On the ubiquity of certain total type structures

Under consideration for publication in Math. Struct. in Comp. Science

On the ubiquity of certain total type structures

John Longley

Received 3 April 2007

It is a fact of experience from the study of higher type computability that a wide range of approaches to defining a class of (hereditarily) total functionals over $\mathbb{N}$ leads in practice to a relatively small handful of distinct type structures. Among these are the type structure C of Kleene-Kreisel *continuous functionals*, its effective substructure $\mathsf{C}^{\mathit{eff}}$, and the type structure HEO of the *hereditarily effective operations*. However, the proofs of the relevant equivalences are often non-trivial, and it is not immediately clear why these particular type structures should arise so ubiquitously.

In this paper we present some new results which go some way towards explaining this phenomenon. Our results show that a large class of *extensional collapse* constructions always give rise to C, $\mathsf{C}^{\mathit{eff}}$ or HEO (as appropriate). We obtain versions of our results for both the "standard" and "modified" extensional collapse constructions. The proofs make essential use of a technique due to Normann.

Many new results, as well as some previously known ones, can be obtained as instances of our theorems, but more importantly, the proofs apply uniformly to a whole family of constructions, and provide strong evidence that the above three type structures are highly canonical mathematical objects.

1. Introduction

This paper presents some new results on computability at higher types. Loosely speaking, we will be working with various collections of *objects of finite type*, in which some set of *type 0 objects* (such as the natural numbers) is taken as basic, and the set of *type* $k+1$ *objects* is some set of "operations" mapping type k objects to type 0 objects. The operations in question might be ordinary mathematical functions, or more intensional objects such as algorithms or programs of some kind.

The general problem of trying to identify reasonable definitions of computability for objects of finite type has a long history, which we have reviewed in our survey paper (Longley 2005a). As argued there, it is far from clear *a priori* what one ought to mean by computability in the higher type setting, but it is a pleasing fact of experience that a wide range of plausible approaches leads in practice to a relatively small handful of distinct notions of computability. For example, if we restrict attention to classes of *functions* (that is, extensional operations) of finite types over the natural numbers, it appears that practically all known plausible definitions lead to one of six classes of functionals, giving us two notions of "total computable functional", and four notions of "partial computable

functional". The situation is summarized at the end of (Longley 2005a), and will be expounded more fully in two sequel papers.

Each of these six notions admits a wide range of different mathematical characterizations, and in each case we have a clutch of theorems asserting that these various characterizations all lead to the same class of functionals. (This is rather analogous to the situation in ordinary first order computability theory, where definitions via Turing machines, lambda calculus, recursion schemes etc. all give rise to the same class of computable first order functions.) In the higher type setting, many of these theorems are non-trivial and surprising, since they often relate constructions which appear totally different in character.

In this paper we will concentrate in particular on certain classes of *hereditarily total* functionals (that is, total functionals acting on total functionals acting on total functionals ... and so on down to ground type). A class of such functionals can be conveniently embodied by a *total type structure* over the set $\mathbb{N}$ of natural numbers (we will define this notion precisely in Section 4). Several natural ways of constructing type structures of this kind are surveyed in (Longley 2005a), along with a number of equivalence results relating such constructions. A consideration of these results leads us to observe that there are three type structures which seem to arise with remarkable frequency:

1 Constructions based on some idea of "continuous functions acting on continuous data" tend to lead consistently to the type structure C of Kreisel's *continuous functionals* (Kreisel 1959), or equivalently Kleene's *countable functionals* (Kleene 1959b).
2 Constructions based on an idea of "effective functions on continuous data" tend to lead to the structure $\mathsf{C}^{\mathit{eff}}$ of *effective continuous functionals*. This can be regarded as an effective substructure of C.
3 Constructions based on an idea of "effective functions on effective data" tend to lead to the type structure HEO of *hereditarily effective operations*, or equivalently to the structure $\mathsf{C}^{\mathit{heff}}$ of *hereditarily effective continuous functionals* (Kreisel 1959; Troelstra 1973).[†]

Since these type structures made their debut in the late 1950s, a large number of equivalent characterizations of them have been obtained, contributing to the impression that these structures are in some sense natural or canonical objects. (We will review some of these results in Section 4.2; see (Longley 2005a, Section 3) for a more historical account.) Note that in the above, the words "continuous" and "effective" can be made precise in various different ways; the point is that in each of the three cases, all reasonable choices lead to the same structure.

The structure C is not itself a candidate for a class of "computable" functions (unless one allows one's "programs" to be infinite objects), but it nevertheless plays an

[†] In much of the earlier literature, the functionals in $\mathsf{C}^{\mathit{eff}}$ are known as the *recursive(ly) continuous functionals*, and those in $\mathsf{C}^{\mathit{heff}}$ are known as the *hereditarily recursive(ly) continuous functionals* (see e.g. (Troelstra 1973; Beeson 1985)). This terminology was followed in (Longley 2005a), where the structures C and $\mathsf{C}^{\mathit{heff}}$ were denoted by RC and HRC respectively. In the present paper, however, our terminology and notation accord with the more modern practice of using the term *recursive* only in contexts where the concept of self-invocation is specifically intended — see (Soare 1999).

important role in the study of higher type computability. The other two type structures, $\mathsf{C}^{\mathit{eff}}$ and HEO, represent the two reasonable notions of "total computable functional" alluded to above.[‡] These two notions are quite different in flavour and are in some sense incomparable: there are functionals in HEO that have no counterpart in $\mathsf{C}^{\mathit{eff}}$, and *vice versa*.

It thus appears that for definitions of a class of total computable functionals, the main dichotomy is between approaches in which functions defined on arbitrary continuous objects of the appropriate type, and approaches in which they are defined on just the computable (*i.e.* effective) objects of the appropriate type. Other aspects of the definitions which one might expect to be significant, such as how we choose to represent functions intensionally and how these intensions are allowed to interact, typically seem not to make any difference to the resulting class of total functionals. Curiously, this is in diametric opposition to the situation for notions of *partial* computable functional, where these other considerations turn out to be critical, but the distinction between "continuous" and "effective" is of minor importance.

What we have sketched here is simply an impression that emerges from a number of individual results connecting up different constructions. It is therefore natural to ask whether one can give some kind of explanation for *why* these three type structures arise so ubiquitously, and whether our observations 1–3 above can be made precise in some way. The purpose of this paper is to present some results which go some way towards realizing these objectives. We will obtain some general theorems saying that *any* construction that follows a certain pattern will result in the type structure C (respectively $\mathsf{C}^{\mathit{eff}}$, HEO).

Our theorems are thus of a somewhat more sweeping character than previous results in the area, and cover many (though not all) of the constructions hitherto considered in the literature as special cases. Hence, our approach provides a unified explanation (and indeed a uniform proof) of several of the previously known equivalence results. In addition, a wealth of new characterizations of these type structures can be obtained as further instances of our theorems. Perhaps most importantly, by virtue of their generality, our results will furnish some kind of mathematical explanation for observations 1–3, and (in the author's view) provide strong confirmation that the structures C, $\mathsf{C}^{\mathit{eff}}$ and HEO are highly canonical mathematical objects.

In view of the generality of our results, we will need some general framework for describing the class of constructions we have in mind. Roughly speaking, our results will cover a large class of constructions that can be naturally presented as instances of the *extensional collapse* construction. In order to articulate our results, we will make use of a framework from realizability involving *typed partial combinatory algebras* (TPCAs), first introduced in (Longley 1999b). As we shall see, many of the known constructions of C

[‡] There is also another important notion of total computable functional, more restrictive than either of these — namely, the one given by Kleene's schemata S1–S9 (Kleene 1959a). In the author's view, this notion is best treated in terms of *partial* functionals so we count it as one of our four partial notions. (This point of view was advocated in (Platek 1966), and we hope to develop it further elsewhere.) We will not say much about S1–S9 computability in this paper, but will briefly allude to how it relates to our present results in Section 10.2).

and $\mathsf{C}^{\mathit{eff}}$, and all known constructions of HEO, are easily seen to be equivalent to constructions that fit within this framework (at least if we also admit *modified extensional collapses*, which we will discuss towards the end of the paper). A plentiful supply of additional constructions is also provided by many of the structures considered in denotational semantics.

On the face of it, the extensional collapse construction does not give a particularly concrete handle on the resulting type structure, and would seem to be very sensitive to the choice of TPCA in question. Indeed, examples drawn from the world of *partial* type structures show that this construction can sometimes exhibit quite capricious behaviour (see the discussion following Remark 4.6), and even in the light of the known equivalence results, there is no particular reason to expect in advance that extensional collapses in the total setting should be any less erratic in general. However, our main theorems will show that all constructions that fit within our framework, and satisfy certain technical conditions, lead to $\mathsf{C}, \mathsf{C}^{\mathit{eff}}$ or HEO as appropriate. The technical conditions turn out to hold in most but not all cases of interest (see Section 10 for a discussion). The proofs of our theorems make essential use of a technique introduced in (Normann 2000), where it is used to show that the total computable functionals arising from the standard Ershov-Scott model are all definable in the language PCF. Normann's proof is already quite technical, and ours are even more so, since several further ingredients need to be added in order to adapt the argument to our more general setting.

The results of this paper were announced in an extended abstract (Longley 2005b). Readers interested in understanding the statements of our results but not the proofs may prefer to consult this shorter version, which takes a somewhat more direct route to the results involving less machinery.[§]

1.1. *Motivations*

For the author, the task of mapping out the various reasonable notions of higher type computability is of intrinsic conceptual interest. However, the results of this paper also have potential repercussions in other areas; here we briefly mention two possible kinds of applications which might be seen as providing further motivation for our inquiry.

One kind of motivation comes from the study of formal systems for constructive logic, such as first order Heyting arithmetic HA or its higher order counterpart HA^ω (see for example (Troelstra 1973) and (Beeson 1985)). For metamathematical purposes, one often considers interpretations of such systems (such as realizability interpretations) that embody some kind of constructive or computational content. Frequently, these interpretations are based on some notion of a constructive or computable operation of finite type. In the case of HA^ω with extensionality, in particular, one is naturally led to consider the

[§] The reader is warned that the published version of (Longley 2005b) contains a serious omission, in that the statements of our main theorems for C and $\mathsf{C}^{\mathit{eff}}$ in (Longley 2005b) lack some technical conditions which appear to be necessary for our proofs. A corrected version of this extended abstract is available from the author's home page. Correct statements of the theorems also appear as Theorems 5.13 and 5.14 below.

class of the hereditarily total functionals that arise from such a notion, so in order to study properties of our interpretation we will surely want to know what this class of functionals is. The results of the present paper will answer this question in a large number of cases, allowing existing knowledge about the resulting class of functionals to be applied to the interpretation in question.

A second motivation comes from computer science. In general (as we have argued in (Longley 2005a, Section 4)), the *partial* notions of higher type computability are both intrinsically better behaved than the total notions and also of greater relevance for computer science. However, there is also some interest in the notion of *totality* in various programming languages, as discussed for example in (Plotkin 1997). Although in principle programming languages allow us to write non-terminating programs, in practice one mostly wants to write programs that do terminate, and these are certainly easier to reason about, since we do not have to worry about the possibility of "undefinedness". It is therefore of interest to consider the class of *total* computable operations that a given programming language supports. As explained in (Plotkin 1997), there are various subtle issues involved in defining a good notion of totality for programs of higher type, but in any case, we will certainly be interested in knowing what class of total operations we obtain from the definitions we adopt. Once again, the results of the present paper will immediately answer this question in many interesting cases.

1.2. *Structure of paper*

The remainder of the paper is structured as follows.

Sections 2 to 5 are devoted to setting up the framework within which we shall work and marshalling all the prerequisites that we need. In Section 2 we present the definition of *typed partial combinatory algebras* (TPCAs) and develop some basic theory for these structures. The material here develops ideas introduced in (Longley 1999b), and is largely a straightforward adaptation of familiar ideas from the untyped case. In Section 3 we illustrate the scope of these ideas with a variety of examples, concentrating mostly on structures that will play important roles in the rest of the paper. In Section 4 we show how TPCAs give rise to total type structures via the (standard) *extensional collapse* construction, and discuss how this fits within the framework of *standard realizability models*. Again, this is mainly a straightforward adaptation of some well-known ideas. We then mention some instances of the extensional collapse construction that have been previously considered in the literature — in this way we introduce the three type structures that will be our main objects of study, and review some of their key properties. In Section 5 we introduce the key ideas of *effective* and *continuous* TPCAs, drawing on our general framework of *applicative morphisms* as considered in (Longley 1995; Longley 1999b). Using these ideas, we are at last able to give precise statements of the main theorems of the paper.

In Section 6 we give a conceptual overview of our proofs of these theorems and their relationship to Normann's work — this short section is in some sense the heart of the paper. We also deal with a minor preliminary issue: the reduction from arbitrary types to pure types. In Sections 7 and 8 we present the detailed technical proofs of our theorems

for the continuous and effective cases respectively. In Section 9 we show that our proof techniques can be adapted to yield analogous results for the *modified extensional collapse* construction (corresponding to modified realizability models).

In Section 10 we mention some applications and specific instances of our results, and tie up a few other loose ends. Finally, in Section 11 we try to assess the significance of our results, point out some of their limitations, and suggest some problems for further research.

1.3. *Notation*

We first fix some notational conventions. We will be considering the *simple types* σ generated by the grammar:¶

$$\sigma \ ::= \ \overline{0} \ \mid \ \sigma_1 \times \sigma_2 \ \mid \ \sigma_1 \to \sigma_2$$

We use $\rho, \sigma, \tau, \upsilon$ to range over types, and consider $\times$ and $\to$ to be right-associative, so that *e.g.* $\rho \to \sigma \to \tau$ means $\rho \to (\sigma \to \tau)$. We inductively define $\overline{n+1}$ to be the type $\overline{n} \to \overline{0}$, and we refer to the types $\overline{n}$ as the *pure types*. For typographical convenience, we often write n in place of $\overline{n}$ when it appears as a subscript or superscript.

If $r > 0$, we write σ^r for the r-fold product type $\sigma \times \cdots \times \sigma$. We also write $\sigma^{(r)} \to \tau$ for the type

$$\underbrace{\sigma \to \cdots \to \sigma}_{r} \to \tau$$

In connection with potentially non-denoting mathematical expressions e, we write $e\!\downarrow$ to mean "e is defined"; $e = e'$ to mean "e, e' are defined and equal" (strict equality); $e \simeq e'$ to mean "if either e or e' is defined then so is the other and they are equal" (Kleene equality); and $e \succeq e'$ to mean "if e' is defined then so is e and they are equal". All other predicate symbols are used in a strict sense, implying that their arguments are defined. Note in particular that $e \neq e'$ means "e, e' are defined and unequal".

We write $f : X \to Y$ [resp. $f : X \rightharpoonup Y$] to mean that f is a (set-theoretic) total [resp. partial] function from X to Y. We also sometimes write $(X \to Y)$ [resp. $(X \rightharpoonup Y)$] for the set of all total [resp. partial] functions from X to Y. We often write $\mathbb{N}^{\mathbb{N}}$ in place of $(\mathbb{N} \to \mathbb{N})$.

When working with finite sequences of any kind, we will follow the computer science convention of indexing the elements by numbers $0, \ldots, n-1$ (for example), where n is some variable giving the length of the sequence. (We allow $x_0, \ldots, x_{n-1}$ to mean the empty sequence when $n = 0$.) Although this leads to a proliferation of '-1's which may appear perverse, the author thinks that this convention comes to seem very pleasant if it is applied consistently.

We write $\mathbb{N}^*$ for the set of finite sequences of natural numbers, and take $\langle \cdots \rangle$ to be

¶ Mathematically it makes little essential difference whether product types are included in our system. Including products tends to add clutter to the basic definitions such as Definition 2.1, but pays off in the long term since it allows later definitions such as 4.4 and 5.1 to be presented in their natural form.

some fixed primitive recursive encoding $\mathbb{N}^* \to \mathbb{N}$, with corresponding primitive recursive operations cons : $\mathbb{N}^2 \to \mathbb{N}$, length : $\mathbb{N} \to \mathbb{N}$, proj : $\mathbb{N}^2 \to \mathbb{N}$ and tail : $\mathbb{N} \to \mathbb{N}$ satisfying

$$\begin{aligned} \text{cons}\,(m, \langle n_0, \ldots, n_{r-1}\rangle) &= \langle m, n_0, \ldots, n_{r-1}\rangle \\ \text{length}\,(\langle n_0, \ldots, n_{r-1}\rangle) &= r \\ \text{proj}\,(i, \langle n_0, \ldots, n_{r-1}\rangle) &= n_i \quad \text{if } i < r \\ \text{tail}\,(\langle n_0, \ldots, n_{r-1}\rangle) &= \langle n_1, \ldots, n_{r-1}\rangle \quad \text{if } r > 0 \\ \text{tail}\,(\langle\rangle) &= \langle\rangle \end{aligned}$$

For any total function $g : \mathbb{N} \to \mathbb{N}$, we write $\widetilde{g} : \mathbb{N} \to \mathbb{N}$ for the *course-of-values* function defined by

$$\widetilde{g}(n) \;=\; \langle g(0), \ldots, g(n-1)\rangle.$$

2. Typed partial combinatory algebras

This section is concerned with setting up the general framework needed to express our results. We present a variant of the definition of *typed partial combinatory algebras* (TPCAs) which we introduced in (Longley 1999b) (under the name of *partial combinatory type structures*). Many of the known constructions giving rise to C, $\mathsf{C}^{\mathit{eff}}$ and HEO involve structures of this kind. In the following section we will give several concrete examples.

Definition 2.1 (TPCA). (i) A *typed partial combinatory algebra* (or *TPCA*) A consists of

— a set A_σ for each type σ;
— a partial function $\cdot_{\sigma\tau} : A_{\sigma\to\tau} \times A_\sigma \rightharpoonup A_\tau$ (known as *application*) for each σ, τ,

such that for any types ρ, σ, τ there exist elements

$$\begin{aligned} \mathbf{k}_{\sigma\tau} &\in A_{\sigma\to\tau\to\sigma} \\ \mathbf{s}_{\rho\sigma\tau} &\in A_{(\rho\to\sigma\to\tau)\to(\rho\to\sigma)\to\rho\to\tau} \\ \mathbf{pair}_{\sigma\tau} &\in A_{\sigma\to\tau\to(\sigma\times\tau)} \\ \mathbf{fst}_{\sigma\tau} &\in A_{(\sigma\times\tau)\to\sigma} \\ \mathbf{snd}_{\sigma\tau} &\in A_{(\sigma\times\tau)\to\tau} \end{aligned}$$

satisfying the following conditions for all elements x, y, z of the appropriate types. (Note that we will frequently omit type subscripts where these can be inferred from the context, and treat '$\cdot$' as left-associative, so that $x \cdot y \cdot z$ means $(x \cdot y) \cdot z$.)

$$\begin{aligned} \mathbf{k} \cdot x \cdot y &= x \\ \mathbf{s} \cdot x \cdot y &\downarrow \\ \mathbf{s} \cdot x \cdot y \cdot z &\succeq (x \cdot z) \cdot (y \cdot z) \\ \mathbf{fst} \cdot (\mathbf{pair} \cdot x \cdot y) &= x \\ \mathbf{snd} \cdot (\mathbf{pair} \cdot x \cdot y) &= y \end{aligned}$$

(ii) A *TPCA with numerals*, or *N-TPCA*, is a TPCA A in which there are elements

$$\begin{array}{rcl} \widehat{0}, \widehat{1}, \widehat{2}, \ldots & \in & A_{\overline{0}} \\ \mathbf{suc} & \in & A_{\overline{0}\to\overline{0}} \\ \mathbf{rec}_\sigma & \in & A_{\sigma\to(\overline{0}\to\sigma\to\sigma)\to\overline{0}\to\sigma} \quad \text{(for each } \sigma\text{)} \end{array}$$

such that for all x, f of appropriate types and all $n \in \mathbb{N}$ we have

$$\begin{array}{rcl} \mathbf{suc} \cdot \widehat{n} & = & \widehat{n+1} \\ \mathbf{rec} \cdot x \cdot f \cdot \widehat{0} & = & x \\ \mathbf{rec} \cdot x \cdot f \cdot \widehat{n+1} & \succeq & f \cdot \widehat{n} \cdot (\mathbf{rec} \cdot x \cdot f \cdot \widehat{n}) \end{array}$$

If e is a long mathematical expression denoting a natural number, we will sometimes write $\widehat{e}$ or $(e)\widehat{\ }$ in place of $\widehat{e}$ when this helps to clarify the span of the $\widehat{\cdots}$ operator.

(iii) A *TPCA with numerals and general recursion*, or *NR-TPCA*, is an N-TPCA A containing elements

$$\mathbf{y}_\rho \quad \in \quad A_{(\rho\to\rho)\to\rho} \quad \text{(for each type } \rho = \sigma \to \tau\text{)}$$

such that for all $f \in A_{\rho\to\rho}$ and $x \in A_\sigma$ we have

$$\begin{array}{rcl} \mathbf{y} \cdot f & \downarrow & \\ \mathbf{y} \cdot f \cdot x & \succeq & f \cdot (\mathbf{y} \cdot f) \cdot x \end{array}$$

Strictly speaking, the choice of elements $\mathbf{k}, \mathbf{s}, \mathbf{pair}, \ldots$ is not part of the data for a TPCA — only the existence of such elements is required. A TPCA together with a suitable choice of the appropriate combinators will be referred to as an *explicit* TPCA (and similarly for N-TPCAs and NR-TPCAs).

Note that our equation for $\mathbf{y}$ is weaker than the more familiar fixed point equation $\mathbf{y} \cdot f = f \cdot (\mathbf{y} \cdot f)$. This extra generality is required for several of the natural examples, such as the PCA K_1 (see Example 3.1). Actually, in the present paper we shall only ever need the combinators $\mathbf{y}_1$ and $\mathbf{y}_2$ (see Section 10.2.2).

In this paper we will take the definition of NR-TPCAs as our basic notion of a "model of higher type computation". As we shall see shortly, any NR-TPCA has enough structure to support "general recursive computation" over the natural numbers.

We will also need an appropriate notion of substructure for TPCAs:

Definition 2.2. A *sub-TPCA* A' of a TPCA A consists of a family of subsets $A'_\sigma \subseteq A_\sigma$ (one for each σ), such that (for some choice of $\mathbf{k}_{\sigma\tau}, \mathbf{s}_{\rho\sigma\tau}, \mathbf{pair}_{\sigma\tau}, \mathbf{fst}_{\sigma\tau}, \mathbf{snd}_{\sigma\tau}$ suitable for A) we have

— $\mathbf{k}_{\sigma\tau}, \mathbf{s}_{\rho\sigma\tau}, \mathbf{pair}_{\sigma\tau}, \mathbf{fst}_{\sigma\tau}, \mathbf{snd}_{\sigma\tau} \in A'$ for all ρ, σ, τ;
— A' is closed under application, in the sense that

$$f \in A'_{\sigma\to\tau} \wedge x \in A'_\sigma \wedge f \cdot x\downarrow \implies f \cdot x \in A'_\tau.$$

Furthermore, if A is an N-TPCA [resp. NR-TPCA] and A' contains elements $\widehat{n}, \mathbf{suc}, \mathbf{rec}_\sigma$ [resp. $\widehat{n}, \mathbf{suc}, \mathbf{rec}_\sigma$ and $\mathbf{y}_\rho$] suitable for A, we say A' is a *sub-N-TPCA* [resp. *sub-NR-TPCA*] of A.

2.1. *A formal language for TPCAs*

Throughout much of this paper we will make use of a formal language for denoting elements of TPCAs. Our language will be based largely on familiar ideas from combinatory logic (see *e.g.* (Barendregt 1984; Beeson 1985)), but incorporates some additional features which will facilitate the construction of complex programs. Firstly, we gain finer control over the evaluation of combinatory expressions by means of what we shall call *protected expressions*. Secondly, the possibility of freely extending our language by adding constants for previously defined programs is built in from the start. Whilst our approach may appear rather formal, experience shows that there are some subtle pitfalls to be avoided in connection with TPCAs, and in our view a secure foundation for programming in TPCAs is essential if we are to conduct our main proofs with any real confidence. Since a clear intuition for how to work with combinators is so crucial for an understanding of the algorithms to be presented in Sections 7 and 8, we include some examples here to illustrate our formal language in action, and to point out some of the pitfalls.

Definition 2.3. Suppose B is some set of *basic constant symbols* b^σ, each with an associated type σ.

Let $\mathcal{L}(B)$ be the set of well-typed *expressions* e, each with an associated type σ, built up inductively as follows:

— Each $b^\sigma \in B$ is an expression of type σ.
— For each type σ we have an infinite supply of *derived constant symbols* c^σ, and each of these is an expression of type σ.
— Likewise, for each type σ we have an infinite supply of *variable symbols* x^σ, and each of these is an expression of type σ.
— Given expressions e and e' of types and σ respectively, we have an expression ee' of type τ.
— Given an expression e of type σ, we have an expression $\lceil e \rceil$ also of type σ.

We write $e : \sigma$ to mean "e is a well-typed expression of type σ". An expression is *closed* if it contains no variable symbols.

We use teletype identifiers (*e.g.* `k`, `f`, `n`) as particular basic constants, derived constants and variables, and as above use b, c, x respectively as metavariables ranging over these classes of symbols. We take juxtaposition to be left-associative, so that $ee'e''$ means $(ee')e''$. Expressions of the form $\lceil e \rceil$ will be called *protected* — the role of such expressions will be explained shortly.

Definition 2.4. (i) A *definition environment* Δ for $\mathcal{L}(B)$ is a list of equations

$$(c_0^{\sigma_0} \equiv e_0,\ \ldots\ , c_{s-1}^{\sigma_{s-1}} \equiv e_{s-1})$$

where the c_i are distinct derived constant symbols, and each e_i is a closed $\mathcal{L}(B)$ expression of type σ_i containing no derived constants apart from $c_0, \ldots, c_{i-1}$. We then write $\mathcal{L}(B)^\Delta_\sigma$ for the set of well-typed expressions $e : \sigma$ such that all derived constants appearing in e are among $c_0, \ldots, c_{s-1}$, and $\mathcal{L}(B)^{\Delta 0}_\sigma$ for the set of closed such expressions. For the case where Δ is empty, we abbreviate these to $\mathcal{L}(B)_\sigma$ and $\mathcal{L}(B)^0_\sigma$ respectively.

(ii) Let B_0 be the set consisting of symbols $\mathtt{k}_{\sigma\tau}, \mathtt{s}_{\rho\sigma\tau}, \mathtt{pair}_{\sigma\tau}, \mathtt{fst}_{\sigma\tau}, \mathtt{snd}_{\sigma\tau}$ for every ρ, σ, τ. Let B_1 consist of all these together with symbols $\mathtt{0}, \mathtt{1}, \ldots, \mathtt{suc}$ and $\mathtt{rec}_\sigma$ for each σ, and let B_2 consist of all the symbols in B_1 together with symbols $\mathtt{y}_\rho$ for each type $\rho = \sigma \to \tau$. All these symbols have associated types as given in Definition 2.1. We now define languages

$$\text{Comb} \;=\; \mathcal{L}(B_0), \qquad \text{NComb} \;=\; \mathcal{L}(B_1), \qquad \text{NRComb} \;=\; \mathcal{L}(B_2)$$

An explicit TPCA will sometimes be referred to as a *model of* Comb; likewise, an explicit N-TPCA will be a model of NComb and an explicit NR-TPCA will be a model of NRComb. In each of the cases $B = B_0, B_1, B_2$, a model A of Comb(B) comes with an interpretation $\ll b^\sigma \gg \in A_\sigma$ for each basic constant $b^\sigma \in B$.

For the rest of this subsection, suppose $\mathcal{L}$ is any one of our three languages Comb, NComb, NRComb.

Definition 2.5. Let A be a model of $\mathcal{L}$. Given a definition environment Δ for $\mathcal{L}$, and a *valuation* ν assigning elements $\nu(x) \in A_\sigma$ to certain variables $x : \sigma$, we define the evident *interpretation function* $\ll - \gg^\Delta_\nu \colon \mathcal{L} \rightharpoonup A$ to be the smallest partial function such that

— $\ll b^\sigma \gg^\Delta_\nu = \ll b^\sigma \gg$ for each basic constant b of $\mathcal{L}$;
— for each equation $c \equiv e$ in Δ, $\ll c \gg^\Delta_\nu = \ll e \gg^\Delta_\nu$ if the right hand side is defined;
— for each variable x, $\ll x \gg^\Delta_\nu = \nu(x)$ if $\nu(x)$ is defined;
— for any $e : \sigma \to \tau$ and $e' : \sigma$, $\ll ee' \gg^\Delta_\nu = \ll e \gg^\Delta_\nu \cdot \ll e' \gg^\Delta_\nu$ if the right hand side is defined;
— for any $e : \sigma$, $\ll \lceil e \rceil \gg^\Delta_\nu = \ll e \gg^\Delta_\nu$ if the right hand side is defined.

Clearly, if $e : \sigma$ and $\ll e \gg^\Delta_\nu$ is defined then $\ll e \gg_\nu \in A_\sigma$. We write $\ll e \gg^\Delta$ for $\ll e \gg^\Delta_\nu$ where ν is empty.

Given any expression $e \in \mathcal{L}^\Delta_\sigma$, we can obviously expand out the derived constant symbols to obtain an expression $e^\dagger \in \text{Comb}^\emptyset_\sigma$, where $\emptyset$ is the empty definition environment. Clearly $e^\dagger$ contains exactly the same variables as e, and for any A and ν we have $\ll e^\dagger \gg^\emptyset_\nu \simeq \ll e \gg^\Delta_\nu$.

A crucial property of TPCAs is that any operation that may be defined by means of a formal expression is also representable within the TPCA itself. We may capture this idea using the following definition. We write $\mathtt{i}_\sigma$ as an abbreviation for the expression $\mathtt{s}_{\sigma(\bar{0}\to\sigma)\sigma}\mathtt{k}_{\sigma(\bar{0}\to\sigma)}\mathtt{k}_{\sigma\bar{0}}$; note that $\ll \mathtt{i} e \gg_\nu \simeq \ll e \gg_\nu$ for any e, ν.

Definition 2.6. Suppose e is any expression of $\mathcal{L}$, and x is a variable symbol of type σ. Then we write $(\lambda^* x.e)$ to denote the $\mathcal{L}$-expression defined by induction on the structure of e as follows.

$$\begin{aligned}
(\lambda^* x.\, x) &= \mathtt{i}_\sigma \\
(\lambda^* x.\, y) &= \mathtt{k}_{\tau\sigma} y \qquad \text{if } y : \tau \text{ is any variable other than } x \\
(\lambda^* x.\, c) &= \mathtt{k}_{\tau\sigma} c \qquad \text{if } c \text{ is a basic or derived constant of type } \tau \\
(\lambda^* x.\, e'e'') &= \mathtt{s}_{\sigma\tau\upsilon}(\lambda^* x.\, e')(\lambda^* x.\, e'') \quad \text{if } e' : \tau \to \upsilon \text{ and } e'' : \tau \\
(\lambda^* x.\, \lceil e \rceil) &= (\lambda^* x.\, e) \;\text{ if the variable } x \text{ appears in } e
\end{aligned}$$

$$(\lambda^* x.\, \lceil e \rceil) \quad = \quad \mathtt{k}_{\tau\sigma} e \qquad \text{if } e : \tau \text{ and } x \text{ does not appear in } e$$

If $x_0, \ldots, x_{r-1}$ are distinct variables, where $r > 1$, we may define

$$(\lambda^* x_0 \ldots x_{r-1}.\, e) = (\lambda^* x_0.\, (\, \cdots\ (\lambda^* x_{r-1}.\, e) \cdots))$$

We emphasize that the λ^* notation is a meta-notation for defining $\mathcal{L}$-expressions, and not a part of the syntax of $\mathcal{L}$-expressions themselves. Thus, in the above equation, the meta-expression $(\lambda^* x_0.\, (\, \cdots\ (\lambda^* x_{r-1}.\, e) \cdots))$ denotes an $\mathcal{L}$-expression which may be obtained by translating the λ^* abstractions "from the inside outwards".

Clearly, if $x : \sigma$ and $e : \tau$, then $(\lambda^* x.e)$ is a well-typed expression of type $\sigma \to \tau$ which contains exactly the same derived constants as e, and exactly the same variables as e except that x itself does not appear in $(\lambda^* x.e)$.

Proposition 2.7 (Combinatory completeness). Let $e : \tau$ be any expression of $\mathcal{L}$, and suppose $x_0^{\sigma_0}, \ldots, x_{r-1}^{\sigma_{r-1}}$ are distinct variables, where $r > 0$. Let V be the set of variables appearing in e, and $V' = V \cup \{x_0, \ldots, x_{r-1}\}$. Then for any model A of $\mathcal{L}$ and any valuation $\nu : V' \to A$ we have

— $\ll(\lambda^* x_0 \ldots x_{r-1}.\, e) x_0 \cdots x_{r-1} \gg_\nu \ \succeq\ \ll e \gg_\nu$

— for any $s < r$, $\ll(\lambda^* x_0 \ldots x_{r-1}.\, e) x_0 \cdots x_{s-1} \gg_\nu \downarrow$ if for every protected subexpression $\lceil e' \rceil$ of e not containing any of $x_s, \ldots, x_{r-1}$, $\ll e' \gg_\nu \downarrow$.

In particular, if e has no protected subexpressions, then for any $s < r$ we have that $\ll(\lambda^* x_0 \ldots x_{r-1}.\, e) x_0 \cdots x_{s-1} \gg_\nu \downarrow$.

Proof. The first property is proved for the case $r = 1$ by a routine induction on the structure of e, bearing in mind that $\ll(\lambda^* x_0.e) x_0 \gg_\nu \simeq \ll(\lambda^* x_0.e) \gg \cdot \nu(x_0)$ and that $\ll e' e'' \gg_\nu \downarrow$ implies $\ll e' \gg_\nu \downarrow$ and $\ll e'' \gg_\nu \downarrow$. We may then establish the first property for $r > 1$ by induction on r as follows:

$$\begin{aligned}
& \ll(\lambda^* x_0 \ldots x_{r-1}.\, e) x_0 \cdots x_{r-1} \gg_\nu \\
\simeq\ & \ll(\lambda^* x_0.(\lambda^* x_1 \ldots x_{r-1}.e)) \cdot x_0 \gg_\nu \cdot \ll x_1 \gg_\nu \cdot \cdots\cdot \ll x_{r-1} \gg_\nu \\
\succeq\ & \ll(\lambda^* x_1 \ldots x_{r-1}.e) \gg_\nu \cdot \ll x_1 \gg_\nu \cdot \cdots\cdot \ll x_{r-1} \gg_\nu \\
\simeq\ & \ll(\lambda^* x_1 \ldots x_{r-1}.e) x_1 \cdots x_r \gg_\nu \ \succeq\ \ll e \gg_\nu
\end{aligned}$$

The second property is proved for the case $s = 0, r = 1$ by an easy induction on the structure of e. We now consider the case $s = 0$, $r > 1$. Let $d = (\lambda^* x_1 \ldots x_{r-1}.e)$. We first note that if d contains a protected subexpression $\lceil e' \rceil$, then the subexpression $\lceil e' \rceil$ also appears within e itself and does not contain any of $x_1, \ldots, x_{r-1}$ (the proof is an easy induction on r). Now suppose $\ll e' \gg_\nu \downarrow$ for every $\lceil e' \rceil$ occurring in e and not containing any of $x_0, \ldots, x_{r-1}$. Then, for any $\lceil e' \rceil$ occurring in d and not containing x_0 we have $\ll e' \gg_\nu \downarrow$, since $\lceil e' \rceil$ also occurs in e and does not contain x_0 or any of $x_1, \ldots, x_n$. Thus, using the special case $s = 0, r = 1$ established above, we have $\ll(\lambda^* x_0.d) \gg_\nu \downarrow$ as required.

The second property for the case $s > 0$ now follows, since by the first property we have

$$\ll(\lambda^* x_0 \ldots x_{r-1}.\, e) x_0 \cdots x_{s-1} \gg_\nu \ \succeq\ \ll(\lambda^* x_s \ldots x_{r-1}.\, e) \gg_\nu$$

The statement for the case of no protected subexpressions is now immediate. □

We will frequently invoke combinatory completeness to define complex formal expressions which it would be impractical to write out in full. As a convention, a defining equation of the form

$$c\, x_0 \ldots x_{r-1} \;\equiv\; e$$

will abbreviate the corresponding definitional equation

$$c \;\equiv\; (\lambda^* x_0 \ldots x_{r-1}.\, e)$$

When interpreting $\mathcal{L}$ expressions in an NR-TPCA, we will in practice omit reference to the definition environment, writing simply $\ll e \gg_\nu$ and understanding all interpretations to be given relative to the "current definitional environment", consisting of all equations of the above form that have been introduced at that point. This will allow us to write cumulative sequences of definitions much as one would do in a programming language such as Haskell or Standard ML.[‖]

To avoid an unsightly proliferation of $\ll\ \gg$ brackets, we shall systematically adopt a convention of using the italic counterparts of teletype identifiers to stand for the elements obtained by interpreting derived constants in a given TPCA A. For example, we can define a predecessor combinator by the equation

$$\mathtt{pre}\ \mathtt{x}^0 \;\equiv\; \mathtt{rec}\ \mathtt{0}\ \mathtt{k}_{00}\ \mathtt{x}$$

and without further apology we may then write *pre* for the element $\ll\mathtt{pre}\gg \in A_1$ (where the implicit definitional environment of course includes the equation just given, and A is determined by the context). In this instance, it is easy to verify that $pre \cdot \widehat{0} = \widehat{0}$ and $pre \cdot \widehat{n+1} = \widehat{n}$.

For constants c defined using combinatory completeness, this convention works well since it is automatic that in any TPCA the interpretation $\ll c \gg$ is defined. Some extra caution is required if we wish to define constants *directly* — that is, via a definitional equation $c \equiv e$ with no parameters — since we have to check that $\ll e \gg$ is defined before we can refer to the italic counterpart of c.

The following example illustrates some further subtleties associated with derived constants.

Example 2.8. Suppose we define

$$\begin{aligned} \mathtt{one} &\;\equiv\; \mathtt{suc}\ \mathtt{0} \\ \mathtt{applyToOne}\ \mathtt{f}^1 &\;\equiv\; \mathtt{f}\ \mathtt{one} \end{aligned}$$

This is equivalent to defining

$$\mathtt{applyToOne}\ \mathtt{f}^1 \;\equiv\; \mathtt{f}\ \lceil \mathtt{suc}\ \mathtt{0} \rceil$$

in the sense that the expanded expression $\mathtt{applyToOne}^\dagger$ in either case is $\mathtt{si}(\mathtt{k}(\mathtt{suc}\ \mathtt{0}))$ (and hence the denotation *applyToOne* in any model is the same in either case). However,

[‖] Usually the implicit "order of definition" here will coincide with the order in which definitions are actually presented in the text. On the rare occasions where we present definitions out of sequence, we will explain what we are doing.

these definitions are *not* equivalent to

$$\mathtt{applyToOne}\ \mathtt{f}^1 \quad \equiv \quad \mathtt{f}\ (\mathtt{suc}\ \mathtt{0})$$

Nor, indeed, does this last definition lead to the same element $applyToOne \in A_2$ as the previous ones in general, since in the latter case, the application of `suc` to `0` is delayed until an argument $f \in A_1$ has been supplied.††

In the present paper, care is often needed over distinctions of this kind. Indeed, the possibility of making such distinctions is one reason why we have introduced the machinery of derived constants into our language. If we simply used `one` as the name of a previously defined *expression* rather than as a constant symbol, the two definitions of `applyToOne` would yield exactly the same formal expression (and hence the same value in any NR-TPCA).

On the other hand, we could have adopted the more familiar practice of regarding combinatory equations as directly defining elements of A rather than formal expressions, and including in our language a constant symbol for each element of A. However, this would make our formal language dependent on a choice of A, and would make it harder to express the idea that the definition of some operation is "uniform in A". For our purposes, a clean separation between the language and its models seems preferable.

We now explain more fully the role of the protected expressions $\lceil e \rceil$ in our language. The purpose of these expressions is to give us some additional control over the evaluation of subexpressions when writing definitions of the above form. As is clear from Definition 2.5, protecting a subexpression e' in an expression e has no effect on the denotation of e in any environment, but it does have an effect on the way $(\lambda^* x_0 \ldots x_{r-1}.\, e)$ is defined. Informally, in the course of the inside-outwards translation of this meta-expression to an expression e' of $\mathcal{L}$, protected subexpressions will be kept together for as long as possible. This means that, conversely, when the corresponding element $\ll e' \gg$ is successively applied to arguments $x_0, \ldots, x_{r-1}$, the protected subexpression will be evaluated as *early* as possible.

Example 2.9. Consider the defining equations

$$\begin{aligned} \mathtt{E}\ \mathtt{x}^0\ \mathtt{f}^1\ \mathtt{g}^1 \quad &\equiv \quad \mathtt{g}\ (\mathtt{f}\ \ \mathtt{x}) \\ \mathtt{E}'\ \mathtt{x}^0\ \mathtt{f}^1\ \mathtt{g}^1 \quad &\equiv \quad \mathtt{g}\ \lceil \mathtt{f}\ \ \mathtt{x} \rceil \end{aligned}$$

The first of these definitions binds `E` to the expression denoted by $(\lambda^*\mathtt{xfg}.\ \mathtt{g(fx)})$. Applying Definition 2.6 to the innermost λ^* abstraction, we see that this is the expression denoted by $(\lambda^*\mathtt{xf}.\ \mathtt{si(s(kf)(kx))})$. (Of course, by expanding the other two λ^* abstractions we could obtain a much longer concrete expression.) Now suppose A is a model of

†† Strictly speaking, throughout the entire paper we will consider NRComb from a static (denotational) rather than a dynamic (operational) perspective, and there is nothing at all in our framework that obliges us to construe evaluation as a "process in time". However, the dynamic view of computation is often the more intuitive, and we shall frequently appeal to dynamic concepts for motivational purposes. For example, we will sometimes say (somewhat loosely) "e evaluates to $\widehat{n}$", where strictly speaking it would suffice to say "$e = \widehat{n}$".

$\mathcal{L}$ and $x \in A_0$, $f \in A_1$. Then by Proposition 2.7 we have

$$E \cdot x \cdot f \ \succeq \ \ll \mathtt{si}(\mathtt{s}(\mathtt{kf})(\mathtt{kx}))\gg_\nu$$

where $\nu(\mathtt{x}) = x$ and $\nu(\mathtt{f}) = f$. Note that the expression appearing on the right hand side does not involve an application of $\mathtt{f}$ to $\mathtt{x}$; in fact, using the axiom '$\mathbf{s} \cdot x \cdot y\!\downarrow$' from Definition 2.1, we see that $E \cdot x \cdot f$ is always defined, even if $f \cdot x$ is undefined. Of course, in this case, $E \cdot x \cdot f \cdot g$ will typically be undefined for any $g \in A_1$.[‡‡] Intuitively, the application of f to x is "delayed" until the argument g is supplied.

By contrast, the second definition binds $\mathtt{E}'$ to the expression denoted by $(\lambda^*\mathtt{xf}.\ \mathtt{si}(\mathtt{fx}))$. In this case, we have

$$E' \cdot x \cdot f \ \succeq \ \ll \mathtt{si}(\mathtt{fx})\gg_\nu$$

and so $E' \cdot x \cdot f$ is typically undefined if $f \cdot x$ is undefined; intuitively, the application of f to x occurs as soon as both x and f are available. However, this "disadvantage" of $\mathtt{E}'$ over $\mathtt{E}$ is offset by an important advantage: if $f \cdot x = f' \cdot x' = y \in A_0$, then $E' \cdot x \cdot f$ and $E' \cdot x' \cdot f'$ will denote exactly the same element $\mathbf{s} \cdot \mathbf{i} \cdot y \in A_1$ (where $\mathbf{i} = \ll \mathtt{i} \gg$). This is not the case for $\mathtt{E}$, since in general the element $E \cdot x \cdot f = \ll \mathtt{si}(\mathtt{s}(\mathtt{kf})(\mathtt{kx}))\gg_\nu$ will depend on the particular elements x and f chosen.

2.2. *Some representable operations*

We now collect some useful examples of operations that are representable in any N-TPCA [resp. NR-TPCA]. We will leave the explicit construction of these operations using the languages NComb, NRComb as a simple exercise.

Firstly, we will often wish to represent operations involving booleans. We will adopt the convention that 0 represents "true", and any number $n \geq 1$ represents "false". Accordingly, in any N-TPCA A, we take $\mathcal{T} = \{\widehat{0}\}$ and $\mathcal{F} = \{\widehat{1}, \widehat{2}, \ldots\}$; For convenience we also define $\mathtt{tt} \equiv \mathtt{0}$ and $\mathtt{ff} \equiv \mathtt{1}$, so that in any N-TPCA we have $tt = \widehat{0}$ and $ff = \widehat{1}$.

It is straightforward to define NComb constants $\mathtt{if}_\sigma : \overline{0} \to \sigma \to \sigma \to \sigma$ such that, in any N-TPCA A, for $x \in A_0$ and $y, z \in A_\sigma$ we have

$$\begin{aligned} \mathit{if}_\sigma \cdot x \cdot y \cdot z &= y \quad \text{if } x \in \mathcal{T} \\ \mathit{if}_\sigma \cdot x \cdot y \cdot z &= z \quad \text{if } x \in \mathcal{F} \end{aligned}$$

Note that the expression $\mathtt{if}_\sigma\ e\ e_0\ e_1$ will be defined in A only if e, e_0, e_1 are *all* defined. However, in practice we will often wish to write a conditional expression that is defined whenever $e \in \mathcal{T} \cup \mathcal{F}$ and the chosen branch (e_0 or e_1 as appropriate) is defined, regardless of whether the other branch is defined.

We may achieve this effect using a trick known in computer science as *thunking*. That is, we represent a potentially undefined expression $e_i : \sigma$ by means of an expression $t_i : \overline{0} \to \sigma$ (called a *thunk*) which is itself guaranteed to be defined, and from which the value of e_i (if any) may be extracted by applying t to some dummy argument (say $\widehat{0}$).

[‡‡] The appearance of $\succeq$ in Proposition 2.7 allows for the possibility that $E \cdot x \cdot f \cdot g$ *might* be defined, but the point is that we have no right to expect it to be.

We may then apply the conditional $\mathtt{if}_{\overline{0}\to\sigma}$ to suitable thunks $t_0, t_1 : \overline{0} \to \sigma$ representing the branches of the conditional. The thunk for the selected branch can then be applied to the dummy argument; in this way the evaluation of a branch is delayed until after the branch is chosen.

Example 2.10. Suppose we have some derived constant $\mathtt{g} : \overline{1}$ such that $g \cdot \widehat{n+1}$ is known to be defined in A for each n, but $g \cdot \widehat{0}$ is not known to be defined. It is natural to want to check whether a given argument x is non-zero before passing x to g: One might first try to define a "safe" version $\mathtt{g}'$ of $\mathtt{g}$ by

$$\mathtt{g}'\ \mathtt{x}^0 \quad \equiv \quad \mathtt{if}_0\ \mathtt{x}\ \mathtt{0}\ (\mathtt{g}\ \mathtt{x})$$

However, this will not have the desired effect, since $\mathit{if}_0 \cdot \widehat{0} \cdot \widehat{0} \cdot (g \cdot \widehat{0})$ is only guaranteed to be defined if $g \cdot \widehat{0}$ is defined.

The problem can be solved by defining

$$\begin{aligned} \mathtt{thunk}\ \mathtt{f}^1\ \mathtt{x}^0\ \mathtt{z}^0 \quad &\equiv \quad \mathtt{f}\ \mathtt{x} \\ \mathtt{g}'\ \mathtt{x} \quad &\equiv \quad \mathtt{if}_{\overline{0}\to\overline{0}}\ \mathtt{x}\ (\mathtt{k}\ \mathtt{0})(\mathtt{thunk}\ \mathtt{g}\ \mathtt{x})\ \mathtt{0} \end{aligned}$$

To see this at work, we note that

$$\mathtt{thunk} \ \equiv \ (\lambda^*\mathtt{fx}.\ \mathtt{s}(\mathtt{kf})(\mathtt{kx}))$$

so that in A we have

$$\mathit{thunk} \cdot g \cdot \widehat{0} \ \succeq \ \mathbf{s} \cdot (\mathbf{k} \cdot g) \cdot (\mathbf{k} \cdot 0)$$

and in particular $\mathit{thunk} \cdot g \cdot \widehat{0}\downarrow$. Hence

$$\begin{aligned} g' \cdot \widehat{0} \quad &\succeq \quad \mathit{if}_{\overline{0}\to\overline{0}} \cdot \widehat{0} \cdot (\mathbf{k} \cdot \widehat{0}) \cdot (\mathit{thunk} \cdot g \cdot \widehat{0}) \cdot \widehat{0} \\ &\succeq \quad (\mathbf{k} \cdot \widehat{0}) \cdot \widehat{0} \\ &\succeq \quad \widehat{0} \end{aligned}$$

Again, care will often be needed over such issues in the present paper, since the issue of "evaluation time" for potentially undefined expressions is often a delicate one.

Using the constant $\mathtt{if}_0$, we may define constants corresponding to the familiar boolean operators as follows:

$$\begin{aligned} \mathtt{and}\ \mathtt{x}^0\ \mathtt{y}^0 \quad &\equiv \quad \mathtt{if}_0\ \mathtt{x}\ \mathtt{y}\ \mathtt{ff} \\ \mathtt{or}\ \mathtt{x}^0\ \mathtt{y}^0 \quad &\equiv \quad \mathtt{if}_0\ \mathtt{x}\ \mathtt{tt}\ \mathtt{y} \\ \mathtt{not}\ \mathtt{x}^0 \quad &\equiv \quad \mathtt{if}_0\ \mathtt{x}\ \mathtt{ff}\ \mathtt{tt} \end{aligned}$$

It is also easy to define constants $\mathtt{eq}, \mathtt{leq} : \overline{0}^{(2)} \to \overline{0}$ which represent the relations $=, \leq$ on natural numbers relative to the above representation of the booleans.

Next we consider which first order numerical functions we can represent. For $r \geq 1$, let us say a partial function $\phi : \mathbb{N}^r \rightharpoonup \mathbb{N}$ is *represented* in an N-TPCA A by an element $\dot{\phi} \in A_{\overline{0}^r\to\overline{0}}$ if for all $n_0, \ldots, n_{r-1} \in \mathbb{N}$ we have

$$\begin{aligned} \dot{\phi} \cdot \widehat{n_0} \cdot \cdots \cdot \widehat{n_{r-2}} \quad &\downarrow \\ \dot{\phi} \cdot \widehat{n_0} \cdot \cdots \cdot \widehat{n_{r-1}} \quad &\succeq \quad (\phi(n_0, \ldots, n_{r-1}))\widehat{\ } \end{aligned}$$

(Note that if $\phi(n_0, \ldots, n_{r-1})$ is undefined, nothing is said about the value, if any, of $\dot{\phi} \cdot \widehat{n_0} \cdot \cdots \cdot \widehat{n_{r-1}}$.) If $\mathcal{L}$ is either NComb or NRComb, we say an expression e of $\mathcal{L}$ *uniformly represents* $\phi : \mathbb{N}^r \rightharpoonup \mathbb{N}$ if $\ll e \gg$ represents ϕ in every model of $\mathcal{L}$; we also say ϕ is *uniformly representable in* $\mathcal{L}$ if there is some $\mathcal{L}$-expression e with this property.

We have already seen that the functions uniformly representable in any N-TPCA include the usual repertoire of basic functions, and it is easy to show that they are also closed under composition and primitive recursion. Furthermore, using the combinator $\mathbf{y}_1$, it is easy to define a constant $\mathtt{min} : \overline{2}$ corresponding to the *minimization operator*, so that in any NR-TPCA we have

$$min \cdot f = \widehat{n} \quad \text{if} \quad f \cdot \widehat{0}, \ldots, f \cdot \widehat{n-1} \in \mathcal{F} \text{ and } f \cdot \widehat{n} \in \mathcal{T}$$

(We do not require the converse.) In view of these facts, we have:

Proposition 2.11. (i) Every primitive recursive function $\mathbb{N}^r \to \mathbb{N}$ is uniformly representable in NComb.

(ii) Every partial computable function $\mathbb{N}^r \rightharpoonup \mathbb{N}$ is uniformly representable in NRComb.

As an example, we may introduce a derived constant $\mathtt{proj} : \overline{0} \to \overline{0} \to \overline{0}$ of NComb which uniformly represents the primitive recursive function proj from Section 1.3. As a mild convenience, we also introduce further constants

$$\mathtt{proj0} \equiv \mathtt{proj\ 0}, \quad \mathtt{proj1} \equiv \mathtt{proj\ 1}, \quad \mathtt{proj2} \equiv \mathtt{proj\ 2}$$

We may also introduce a derived NComb constant $\mathtt{make\text{-}pair} : \overline{0} \to \overline{0} \to \overline{0}$ of NComb which uniformly represents the primitive recursive function $(n_0, n_1) \mapsto \langle n_0, n_1 \rangle$. In any N-TPCA A, we therefore have elements $\mathit{proj},\ \mathit{make\text{-}pair} \in A_{\overline{0} \to \overline{0} \to \overline{0}}$ and $\mathit{proj0}, \mathit{proj1}, \mathit{proj2} \in A_{\overline{0} \to \overline{0}}$.

We now consider the representation of second order operations in an NR-TPCA. Given a total function $\alpha : \mathbb{N} \to \mathbb{N}$, a partial function $\phi : \mathbb{N}^r \rightharpoonup \mathbb{N}$ is *partial computable uniformly in* α if ϕ can be defined from α plus the basic functions via composition, primitive recursion and minimization. Clearly, any definition of such a function by these means can be regarded as specifying a function $\mathbb{N}^{\mathbb{N}} \to (\mathbb{N}^r \rightharpoonup \mathbb{N})$, by allowing α to vary over $\mathbb{N}^{\mathbb{N}}$. The functions $F : \mathbb{N}^{\mathbb{N}} \to (\mathbb{N}^r \rightharpoonup \mathbb{N})$ obtained in this way will be called the *type 2 partial computable functionals*; they were first described in (Kleene 1952).

From our discussion of partial computable functions, it is clear that any definition of a partial function $\phi : \mathbb{N}^r \rightharpoonup \mathbb{N}$ from a function $\alpha \in \mathbb{N}^{\mathbb{N}}$ by means of composition, primitive recursion and minimization can be translated into a Comb expression $e : \overline{0}^{(r)} \to \overline{0}$ involving (at most) a single variable $\mathtt{a}^1$, such that if $\nu(\mathtt{a})$ represents α then $\ll e \gg_\nu$ represents ϕ. Allowing α now to vary over $\mathbb{N}^{\mathbb{N}}$, by combinatory completeness we may then obtain an element $\dot{F} \in A$ such that for all $\dot{\alpha} \in A_1$, $\dot{F} \cdot \dot{\alpha} \succeq \ll e \gg_\nu$ whenever $\nu(\mathtt{a}) = \dot{\alpha}$. We thus have:

Proposition 2.12. Every type 2 partial computable functional $F : \mathbb{N}^{\mathbb{N}} \to (\mathbb{N}^r \rightharpoonup \mathbb{N})$ is uniformly representable in NRComb, in the sense that there is an NRComb expression $e : \overline{1} \to \overline{0}^r \to \overline{0}$ such that in any NR-TPCA A, for any $\dot{\alpha} \in A_1$ representing any $\alpha \in \mathbb{N}^{\mathbb{N}}$,

and for any $n_0, \ldots, n_{r-1}$, we have

$$\ll e \gg \cdot \dot{\alpha} \cdot \widehat{n_0} \cdots \widehat{n_{r-1}} \ \succeq \ (F(\alpha)(n_0, \ldots, n_{r-1}))\widehat{}$$

Finally, we introduce an example of a type 2 operator that will be useful later. We may define an NComb constant

$$\texttt{all-in-list} : \overline{1} \to \overline{0} \to \overline{0}$$

for testing whether a given predicate holds for all numbers in some finite sequence. More precisely, in any N-TPCA, `all-in-list` will denote an element *all-in-list* such that for any $n_0, \ldots, n_{r-1} \in \mathbb{N}$ and $f \in A_1$ we have

$$\begin{array}{lcll} \textit{all-in-list} \cdot f \cdot \langle n_0, \ldots, n_{r-1} \rangle\widehat{} & = & \textit{tt} & \text{if } f \cdot \widehat{n_i} \in \mathcal{T} \text{ for all } i < r \\ \textit{all-in-list} \cdot f \cdot \langle n_0, \ldots, n_{r-1} \rangle\widehat{} & = & \textit{ff} & \text{if for some } i < r,\ f \cdot \widehat{n_0}, \ldots, f \cdot \widehat{n_{i-1}} \in \mathcal{T} \\ & & & \text{but } f \cdot \widehat{n_i} \in \mathcal{F} \end{array}$$

The existence of a suitable definition for `all-in-list` is not strictly an instance of the foregoing proposition, since f here is not constrained to represent a total function α (and also because we here wish to use NComb rather than NRComb). Nevertheless, it is easy to see how a suitable definition of `all-in-list` may be given in NComb, since the functions associated with the encoding $\langle \cdots \rangle$ in Section 1.3 are all primitive recursive.

3. Examples of TPCAs

We now illustrate our general framework by means of some examples. These serve both to indicate the scope of the above definitions and to introduce some particular TPCAs that will play a role later on. In the following section we will review some known results concerning the extensional collapses of these models.

3.1. *Untyped PCAs*

Any ordinary (untyped) partial combinatory algebra can be viewed as a TPCA. Recall that a partial combinatory algebra (or PCA) consists of a set A equipped with a partial function $\cdot : A \times A \rightharpoonup A$, such that there exist elements $\mathbf{k}, \mathbf{s} \in A$ satisfying the first three conditions given in Definition 2.1. We may regard any PCA A as an NR-TPCA, by setting $A_\sigma = A$ and $\cdot_{\sigma,\tau} = \cdot$ for all σ, τ. In the untyped setting, suitable pairing and projection combinators, a system of numerals and a recursor $\mathbf{y}$ can be constructed from $\mathbf{k}, \mathbf{s}$ by well-known means (see *e.g.* (Barendregt 1984; Longley 1995)).[§§]

Furthermore, we may say A' is a *sub-PCA* of A if $A' \subseteq A$ contains some $\mathbf{k}, \mathbf{s}$ suitable for A and is closed under application in the sense of Definition 2.2. Clearly, if A' is a sub-PCA of A, then considering A, A' as typed structures, A' is a sub-NR-TPCA of A.

[§§] In many expositions, including (Longley 1995), the definition of PCA includes the axiom $\mathbf{s} \cdot x \cdot y \cdot z \simeq (x \cdot z) \cdot (y \cdot z)$. However, the weaker definition using $\succeq$ will suffice for our purposes; moreover, the constructions of suitable combinators $\mathbf{pair}, \mathbf{fst}, \mathbf{snd}, \widehat{n}, \mathbf{rec}, \mathbf{y}$ given in (Longley 1995, Chapter 1) also work in the present setting (note that $\mathbf{y}$ is called Z in *op. cit.*).

The following two examples are particularly important for our purposes:

Example 3.1. *Kleene's first model*, denoted K_1. Here the underlying set is $\mathbb{N}$, and application is defined by $m \bullet n \simeq \phi_m(n)$, where $\phi_0, \phi_1, \ldots$ is some fixed effective enumeration of the unary partial computable functions. (The symbol $\bullet$ will be reserved for application in K_1.) The existence of suitable combinators $\mathbf{k}, \mathbf{s}$ in K_1 is an easy consequence of the S-m-n theorem. Although a system of numerals in K_1 exists for general reasons, it is of course more convenient (and equivalent for our purposes) to take $\widehat{n} = n$.

Example 3.2. *Kleene's second model*, denoted K_2. Here the underlying set is $\mathbb{N}^{\mathbb{N}}$, the set of all total functions $\mathbb{N} \to \mathbb{N}$. To define application in K_2, we need some auxiliary notation. Define a partial function $- \mid - : \mathbb{N}^{\mathbb{N}} \times \mathbb{N}^{\mathbb{N}} \rightharpoonup \mathbb{N}$ by

$$f \mid g \;\simeq\; f(\widetilde{g}(r)) - 1, \text{where } r \text{ is the least number such that } f(\widetilde{g}(r)) > 0.$$

where $\widetilde{g}$ is the course-of-values function defined in Section 1.3. (For the motivation behind this definition, see *e.g.* (Longley 2005a, Section 3.3).) Next, given $g \in \mathbb{N}^{\mathbb{N}}$ and $m \in \mathbb{N}$, we may define $g_m \in \mathbb{N}^{\mathbb{N}}$ by $g_m(0) = m$, $g_m(n+1) = g(n)$. Thus, from any f, g we may obtain a *partial* function $\Lambda m.\ f \mid g_m : \mathbb{N} \rightharpoonup \mathbb{N}$. (We use the meta-lambda notation $\Lambda x.e$ to mean the partial function h defined by $\forall x.\ h(x) \simeq e$.) The application operation for K_2 is now defined by

$$f \odot g \;=\; \begin{cases} \Lambda m.\ f \mid g_m & \text{if this is a total function,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

(The symbol $\odot$ will be reserved for application in K_2.) The functions $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ that are representable in K_2 (*i.e.* that are of the form $\Lambda g.\ f \odot g$ for some $f \in K_2$) are precisely the *continuous* functions with respect to the Baire topology on $\mathbb{N}^{\mathbb{N}}$.

A curious feature of K_2 is that the definedness of application is not semidecidable: more precisely, the set $\{(f, g) \mid f \odot g\!\downarrow\}$ is not open in the Baire topology, because of the infinitary nature of the condition that the function $\Lambda m.\ f \mid g_m$ is total. This curiosity will have mild repercussions in Section 5.4.

The structure $(K_2, \odot)$ can be shown to be a PCA — see *e.g.* (Kleene and Vesley 1965). A convenient choice of numerals $\widehat{k}$ in K_2 may be given by defining $\widehat{k}(0) = k$, $\widehat{k}(j+1) = 0$.

Moreover, by restricting to the total *computable* functions $\mathbb{N} \to \mathbb{N}$, we obtain a sub-PCA K_2^{eff} of K_2. (In this paper we will uniformly use the superscript $^{\mathit{eff}}$ to indicate an effective substructure of some given structure.)

Other PCAs, whose definition will not be required in this paper, include: Scott's D_∞ models (Scott 1972); his *graph model* $\mathcal{P}\omega$ (Scott 1976); Plotkin's $\mathbb{T}^\omega$ (Plotkin 1978); van Oosten's $\mathcal{B}$ (van Oosten 1999); the effective sub-PCAs of all these; and various term models of untyped lambda calculi, such as Λ^0/β, the set of closed lambda terms modulo β-equivalence (Barendregt 1984). Several of these PCAs are discussed *e.g.* in (Longley 1999a).

3.2. *CCC models*

Many cartesian closed categories naturally give rise to interesting TPCAs. Start with a cartesian closed category $\mathcal{C}$, with a chosen object $\bar{N}$. Define objects $\mathbf{A}_\sigma$ by

$$\mathbf{A}_0 \;=\; \bar{N}, \qquad \mathbf{A}_{\sigma\times\tau} \;=\; \mathbf{A}_\sigma \times \mathbf{A}_\tau, \qquad \mathbf{A}_{\sigma\to\tau} \;=\; \mathbf{A}_\tau^{\mathbf{A}_\sigma}.$$

and consider the family of sets $A_\sigma = \mathrm{Hom}(1, \mathbf{A}_\sigma)$, with total functions $\cdot_{\sigma\tau}$ induced by the evaluation morphisms $\mathbf{A}_\tau^{\mathbf{A}_\sigma} \times \mathbf{A}_\sigma \to \mathbf{A}_\tau$. It is easy to check that these data constitute a total TPCA A; the existence of suitable $\mathbf{k}, \mathbf{s}, \mathbf{pair}, \mathbf{fst}, \mathbf{snd}$ follows from the fact that $\mathcal{C}$ is cartesian closed.

Now suppose $\bar{N}$ comes equipped with an inclusion $\mathbb{N} \hookrightarrow \mathrm{Hom}(1, \bar{N})$, and $\mathcal{C}$ contains successor and recursor morphisms in an obvious sense. (This will be true if $\bar{N}$ is a natural number object in $\mathcal{C}$, for instance.) Then A will contain suitable elements $\mathbf{suc}, \mathbf{rec}$ and will hence be an N-TPCA. Finally, if $\mathcal{C}$ contains a suitable "fixed point" morphism in an obvious sense, then A will be an NR-TPCA.

Many categories based on order-theoretic structures (*e.g.* continuous lattices (Scott 1972), stable domains (Berry 1978)) give rise to NR-TPCAs in this way, as do many "intensional" categories such as the Berry-Curien *sequential algorithms* model (Berry and Curien 1982), and many of the categories of games considered by Abramsky, Hyland *et al* (Abramsky and McCusker 1999; Hyland 1997). (Loosely speaking, any category that provides a "model of PCF" will yield an NR-TPCA.) Typically, in all these models, we take $\bar{N}$ to be the object playing the role of the domain $N_\perp$. In addition, many of these models have evident "effective subcategories" which give rise to sub-TPCAs.

A further class of examples (not technically required for this paper) may be obtained by considering *monads* on any of the above categories, such as the monads corresponding to various computational effects as considered in (Moggi 1991). If $\mathcal{C}$ is a CCC with chosen object $\bar{N}$, and (T, η, μ) is a commutative strong monad on $\mathcal{C}$ such that each η_X is mono, we may define objects $\mathbf{B}_\sigma$ by

$$\mathbf{B}_0 \;=\; \bar{N}, \qquad \mathbf{B}_{\sigma\times\tau} \;=\; \mathbf{B}_\sigma \times \mathbf{B}_\tau, \qquad \mathbf{B}_{\sigma\to\tau} \;=\; (T\mathbf{B}_\tau)^{\mathbf{B}_\sigma}.$$

These come equipped with evaluation morphisms $\epsilon_{\sigma\tau} : \mathbf{B}_{\sigma\to\tau} \times \mathbf{B}_\sigma \to T\mathbf{B}_\tau$. Now consider the family of sets $B_\sigma = \mathrm{Hom}(1, \mathbf{B}_\sigma)$, with *partial* application functions $\cdot_{\sigma\tau}$ defined by

$$f \cdot_{\sigma\tau} x \;=\; y \;\text{ iff }\; \epsilon_{\sigma\tau} \circ \langle f, x\rangle = \eta \circ y$$

One can check that these data constitute a TPCA B; again, this will be an NR-TPCA if $\mathcal{C}$ contains suitable successor, recursor and fixed point morphisms.

3.3. *The partial continuous functionals*

One very important example of a TPCA arising from a CCC is the so-called "Ershov-Scott model" for the simple types — that is, the TPCA of partial continuous functions over $N_\perp$ (Scott 1969; Ershov 1972a). We will write P for this TPCA, and $\mathsf{P}^{\mathit{eff}}$ for its effective submodel. Since these structures will play such a major role in this paper, we will review the definitions of these structures and their basic properties here. The material of this subsection is mostly standard domain theory (see *e.g.* (Stoltenberg-Hansen *et al.*

1994; Amadio and Curien 1998) for further details), but we shall also take the opportunity to fix on some notational conventions of our own which we shall need later.

Given a poset $(X, \sqsubseteq)$, a subset $D \subseteq X$ is called *directed* if it is non-empty and for all $x, y \in D$ there exists $z \in D$ with $x \sqsubseteq z$, $y \sqsubseteq z$. We say $(X, \sqsubseteq)$ is a *directed-complete partial order*, or DCPO, if every directed $D \subseteq X$ has a supremum (*i.e.* a least upper bound) in X, written $\bigsqcup D$. We write $N_\perp$ for the DCPO $\mathbb{N} \sqcup \{\perp\}$ with the ordering $\sqsubseteq_{N_\perp}$ defined by

$$x \sqsubseteq_{N_\perp} y \;\text{ iff }\; x = \perp \vee x = y$$

Suppose $(X, \sqsubseteq_X)$ and $(Y, \sqsubseteq_Y)$ are DCPOs. A function $f : X \to Y$ is *continuous* if whenever $D \subseteq X$ is directed, the set $f(D) \subseteq Y$ is directed and $\bigsqcup f(D) = f(\bigsqcup D)$. Clearly, any continuous function f is also *monotone*: that is, $x \sqsubseteq_X x'$ implies $f(x) \sqsubseteq_Y f(x')$. Let $[X \Rightarrow Y]$ denote the set of continuous functions from X to Y, and let $\sqsubseteq_Y^{pt}$ denote the *pointwise* ordering on $[X \Rightarrow Y]$ defined by

$$f \sqsubseteq_Y^{pt} g \;\text{ iff }\; \forall x \in X.\, f(x) \sqsubseteq_Y g(x)$$

It is routine to check that $([X \Rightarrow Y], \sqsubseteq_Y^{pt})$ is itself a DCPO; indeed, DCPOs and continuous functions form a cartesian closed category in which the exponential $(Y, \sqsubseteq_Y)^{(X, \sqsubseteq_X)}$ may be taken to be $([X \Rightarrow Y], \sqsubseteq_Y^{pt})$.

We may now define a DCPO $(\mathsf{P}_\sigma, \sqsubseteq_\sigma)$ for each type σ by induction as follows:

$$\begin{aligned} (\mathsf{P}_0, \sqsubseteq_0) &= (N_\perp, \sqsubseteq_{N_\perp}), \\ (\mathsf{P}_{\sigma\times\tau}, \sqsubseteq_{\sigma\times\tau}) &= (\mathsf{P}_\sigma \times \mathsf{P}_\tau, \sqsubseteq_\sigma \times \sqsubseteq_\tau), \\ (\mathsf{P}_{\sigma\to\tau}, \sqsubseteq_{\sigma\to\tau}) &= (\mathsf{P}_\tau, \sqsubseteq_\tau)^{(\mathsf{P}_\sigma, \sqsubseteq_\sigma)} \end{aligned}$$

The sets P_σ constitute an NR-TPCA which we shall call P (note that application in P is true function application). We will use Gothic identifiers (*e.g.* $\mathfrak{a}, \mathfrak{f}, \mathfrak{u}, \mathfrak{G}$) to range over elements of P and write $\perp_\sigma$ for the evident least element of P_σ. We say that $\mathfrak{u}, \mathfrak{v} \in \mathsf{P}_\sigma$ are *consistent* if there exists $\mathfrak{w} \in \mathsf{P}_\sigma$ with $\mathfrak{u} \sqsubseteq \mathfrak{w}$, $\mathfrak{v} \sqsubseteq \mathfrak{w}$ (we will often omit type annotations when they can be inferred from the context).

An element $\mathfrak{a}$ of P_σ is called *compact* if whenever $D \subseteq \mathsf{P}_\sigma$ is directed and $\mathfrak{a} \sqsubseteq \bigsqcup D$, there is some $\mathfrak{w} \in D$ such that $\mathfrak{a} \sqsubseteq \mathfrak{w}$. We write P_σ^{comp} for the set of compact elements of $(\mathsf{P}_\sigma, \sqsubseteq_\sigma)$, and given $\mathfrak{u} \in \mathsf{P}_\sigma$, we write $C_\mathfrak{u}$ for the set $\{\mathfrak{c} \in \mathsf{P}_\sigma^{comp} \mid \mathfrak{c} \sqsubseteq \mathfrak{u}\}$.

It can be shown that the DCPOs $(\mathsf{P}_\sigma, \sqsubseteq_\sigma)$ all enjoy the following pleasant properties:

— The set P_σ^{comp} is countable.
— $(\mathsf{P}_\sigma, \sqsubseteq_\sigma)$ is *algebraic*: for any $\mathfrak{u} \in \mathsf{P}_\sigma$, the set $C_\mathfrak{u}$ is directed and $\mathfrak{u} = \bigsqcup C_\mathfrak{u}$.
— $(\mathsf{P}_\sigma, \sqsubseteq_\sigma)$ is *coherent*: a set $A \subseteq \mathsf{P}_\sigma$ has a supremum $\bigsqcup A$ in P_σ iff the elements of A are pairwise consistent.

As a special case of the last condition, we have that if $\mathfrak{u}, \mathfrak{v} \in \mathsf{P}_\sigma$ are consistent then they have a supremum $\mathfrak{u} \sqcup \mathfrak{v} \in \mathsf{P}_\sigma$.

We can be more explicit about what the compact elements of P_σ are. In P_0, every element is compact, and at product types, the compact elements are precisely those whose components are compact elements. Given $\mathfrak{a} \in \mathsf{P}_\sigma^{comp}$ and $\mathfrak{b} \in \mathsf{P}_\tau^{comp}$, we write

$\mathfrak{a} \Rightarrow \mathfrak{b}$ for the "step function" defined by

$$(\mathfrak{a} \Rightarrow \mathfrak{b})(\mathfrak{u}) = \begin{cases} \mathfrak{b} & \text{if } \mathfrak{a} \sqsubseteq \mathfrak{u}, \\ \bot & \text{otherwise} \end{cases}$$

Then $\mathfrak{a} \Rightarrow \mathfrak{b}$ is a compact element of $\mathsf{P}_{\sigma\to\tau}$, and it can be shown that the compact elements of $\mathsf{P}_{\sigma\to\tau}$ are precisely the finite suprema of such elements that exist in $\mathsf{P}_{\sigma\to\tau}$. Furthermore, a simple criterion for the existence of such a supremum can be given: the supremum $\bigsqcup_{i<r}(\mathfrak{a}_i \Rightarrow \mathfrak{b}_i)$ exists if and only if for all $i, j < r$ such that $\mathfrak{a}_i, \mathfrak{a}_j$ are consistent, $\mathfrak{b}_i, \mathfrak{b}_j$ are also consistent.

This characterization suggests the possibility of an effective enumeration of the compact elements of P. We will fix on an enumeration which is similar in essentials to those given in *e.g.* (Plotkin 1977; Stoltenberg-Hansen *et al.* 1994), but for our purposes it will be convenient to build in a few additional hygiene conditions on our representations of compact elements.

Definition 3.3. Define a function $\zeta_0 : \mathbb{N} \to \mathsf{P}_0^{comp}$ by $\zeta_0(0) = \bot$, $\zeta_0(n+1) = n$. Given partial functions $\zeta_\sigma : \mathbb{N} \rightharpoonup \mathsf{P}_\sigma^{comp}$ and $\zeta_\tau : \mathbb{N} \rightharpoonup \mathsf{P}_\tau^{comp}$, let $\zeta_{\sigma\times\tau} : \mathbb{N} \rightharpoonup \mathsf{P}_{\sigma\times\tau}^{comp}$ be the smallest partial function such that

$$\zeta_{\sigma\times\tau}(n) = (\zeta_\sigma(a), \zeta_\tau(b)) \quad \text{if} \quad n = \langle a, b\rangle \text{ and } \zeta_\sigma(a),\ \zeta_\tau(b) \text{ are defined}$$

and let $\zeta_{\sigma\to\tau} : \mathbb{N} \rightharpoonup \mathsf{P}_{\sigma\to\tau}^{comp}$ be the smallest partial function such that

$$\zeta_{\sigma\to\tau}(n) = \bigsqcup_{i<r}(\zeta_\sigma(a_i) \Rightarrow \zeta_\tau(b_i)) \quad \text{if} \quad \begin{cases} n = \langle\langle a_0, b_0\rangle, \ldots, \langle a_{r-1}, b_{r-1}\rangle\rangle, \\ \text{the } \zeta_\sigma(a_i), \zeta_\tau(b_i) \text{ are all defined,} \\ \text{none of the } \zeta_\tau(b_i) \text{ are } \bot_\tau, \text{ and} \\ \text{the join } \bigsqcup_{i<r}(\zeta_\sigma(a_i) \Rightarrow \zeta_\tau(b_i)) \text{ exists in } \mathsf{P}_{\sigma\to\tau} \end{cases}$$

The following facts can be shown for each type σ: the domain of ζ_σ is primitively recursively decidable; the range of ζ_σ is precisely the set of compact elements of P_σ; the relations "$\zeta_\sigma(m) \sqsubseteq \zeta_\sigma(n)$", "$\zeta_\sigma(m) = \zeta_\sigma(n)$" and "$\zeta_\sigma(m), \zeta_\sigma(n)$ are consistent" are primitively recursively decidable in m, n; and there is a primitive recursive function $\mathrm{join}_\sigma : \mathbb{N}^2 \to \mathbb{N}$ such that $\zeta_\sigma(\mathrm{join}_\sigma(m,n)) = \zeta_\sigma(m) \sqcup \zeta_\sigma(n)$ whenever $\zeta_\sigma(m), \zeta_\sigma(n)$ are consistent. Moreover, for each σ, τ, there is a primitive recursive function $\mathrm{apply}_{\sigma\tau} : \mathbb{N}^2 \to \mathbb{N}$ such that for all $m \in \mathrm{dom}\ \zeta_{\sigma\to\tau}$ and $n \in \mathrm{dom}\ \zeta_\sigma$ we have

$$\zeta_{\sigma\to\tau}(m)(\zeta_\sigma(n)) \ = \ \zeta_\tau(\mathrm{apply}_{\sigma\tau}(m,n)).$$

By definition, the *basic* compact elements of P_0 are the elements of $\mathbb{N}$, the basic compact elements of $\mathsf{P}_{\sigma\times\tau}$ are pairs $(\mathfrak{a}, \mathfrak{b})$ where $\mathfrak{a}, \mathfrak{b}$ are basic compact in $\mathsf{P}_\sigma, \mathsf{P}_\tau$ respectively, and the basic compact elements of $\mathsf{P}_{\sigma\to\tau}$ are those of the form $\mathfrak{a} \Rightarrow \mathfrak{b}$, where $\mathfrak{a}$ is compact in P_σ and $\mathfrak{b}$ is basic compact in P_τ. For any $\mathfrak{u} \in \mathsf{P}_\sigma$, we write $B_\mathfrak{u}$ for the set of basic compact elements $\mathfrak{b} \sqsubseteq \mathfrak{u}$. If $n = \langle\langle a_0, b_0\rangle, \ldots, \langle a_{r-1}, b_{r-1}\rangle\rangle \in \mathrm{dom}\ \zeta_{\sigma\to\tau}$, it is easily shown that $\zeta_{\sigma\to\tau}(n)$ is basic compact iff there is some $i < r$ such that $\zeta_\sigma(a_i) \sqsubseteq \zeta_\sigma(a_j)$ and $\zeta_\tau(b_j) \sqsubseteq \zeta_\tau(b_i)$ for all $j < r$, and moreover $\zeta_\tau(b_i)$ is basic compact; in this case we have $\zeta_{\sigma\to\tau}(n) = \zeta_\sigma(a_i) \Rightarrow \zeta_\tau(b_i)$. It follows that for each σ, the unary relation "$\zeta_\sigma(n)$ is basic compact" is primitively recursively decidable.

We also have a syntactic notion of what it means to be a basic *code* for a compact

element. The basic codes for type $\overline{0}$ are simply the numbers $n+1$, the basic codes for type $\sigma \times \tau$ are the numbers $\langle a, b\rangle$ where a, b are basic codes for types σ, τ respectively, and the basic codes for type $\sigma \to \tau$ are the numbers $\langle\langle a, b\rangle\rangle$ where $a \in \mathrm{dom}\ \zeta_\sigma$ and b is a basic code for type τ. Clearly, if n is a basic code for type σ then $\zeta_\sigma(n)$ is basic compact, and in view of the characterization above, for each σ there is a primitive recursive function $\mathrm{basic}_\sigma : \mathbb{N} \to \mathbb{N}$ such that whenever $\zeta_\sigma(n)$ is basic compact, $\mathrm{basic}_\sigma(n)$ is a basic code for type σ and $\zeta_\sigma(\mathrm{basic}_\sigma(n)) = \zeta_\sigma(n)$.

For the purpose of writing codes for compact elements, we will often write $a \mapsto b$ to mean $\langle a, b\rangle$, so that if a codes a compact element and b codes a basic compact element then $\zeta(a \mapsto b) = \zeta(a) \Rightarrow \zeta(b)$. We will also write $\breve{m}$ for $m + 1$ considered as the ζ_0-code for $m \in \mathsf{P}_0$, and $\bot$ for 0. Together these notations give us a useful way of writing codes: *e.g.* if $m_i, n_i, p \in \mathbb{N}$, the code

$$\langle\langle \breve{m}_0 \mapsto \breve{n}_0, \ldots, \breve{m}_{r-1} \mapsto \breve{n}_{r-1}\rangle \mapsto \breve{p}\rangle$$

codes the type 2 basic compact element $(\bigsqcup_i (m_i \Rightarrow n_i)) \Rightarrow p$.

A set $A \subseteq \mathsf{P}_\sigma^{comp}$ is *computably enumerable (c.e.)* if there is a c.e. set $R_A \subseteq \mathrm{dom}\ \zeta_\sigma$ such that $A = \{\zeta_\sigma(n) \mid n \in R_A\}$. Since the relations "$n \in \mathrm{dom}\ \zeta_\sigma$" and "$\zeta_\sigma(m) = \zeta_\sigma(n)$" are both decidable, this is the same as saying that the relation "$\zeta_\sigma(n) \in A$" is c.e. in n. An element $\mathfrak{u} \in \mathsf{P}_\sigma$ is *effective* if there is a c.e. set $A \subseteq \mathsf{P}_\sigma^{comp}$ with $\bigsqcup A = \mathfrak{u}$. Equivalently, $\mathfrak{u}$ is effective if $C_\mathfrak{u}$ is c.e., or if $B_\mathfrak{u}$ is c.e. We write $\mathsf{P}_\sigma^{\mathit{eff}}$ for the set of effective elements of P_σ; obviously $\mathsf{P}_\sigma^{comp} \subseteq \mathsf{P}_\sigma^{\mathit{eff}}$. One can check that if $\mathfrak{f} \in \mathsf{P}_{\sigma\to\tau}^{\mathit{eff}}$ and $\mathfrak{u} \in \mathsf{P}_\sigma^{\mathit{eff}}$ then $\mathfrak{f}(\mathfrak{u}) \in \mathsf{P}_\tau^{\mathit{eff}}$, and indeed that the sets $\mathsf{P}_\sigma^{\mathit{eff}}$ constitute a sub-NR-TPCA of P. Furthermore, any $\mathfrak{f} \in \mathsf{P}_{\sigma\to\tau}$ is completely determined by its restriction to $\mathsf{P}_\sigma^{\mathit{eff}}$ (or even to P_σ^{comp}), so we may, if we prefer, consider each $\mathsf{P}_{\sigma\to\tau}^{\mathit{eff}}$ to be simply a set of functions from $\mathsf{P}_\sigma^{\mathit{eff}}$ to $\mathsf{P}_\tau^{\mathit{eff}}$.

3.4. *Syntactic models*

Another class of TPCAs are those arising as term models for various syntactic calculi or "programming languages" for operations of finite type, such as PCF (Plotkin 1977) and various extensions thereof. We shall concentrate here on the term model for the language NRComb defined in Section 2.1, and its infinitary counterpart. These models will play an important conceptual role within the present paper, as will be explained in Section 6.1; however, technically these models will not feature in our proofs, so the reader is advised that a quick glance through this section will probably be all that is necessary.

In order to define this model, we introduce a simple calculus of "provable formulae" for NRComb. The formulae of our calculus will be of two kinds:

— If e is a closed term of NRComb in the empty definition environment, then $e\!\downarrow$ is a formula.

— If e, e' are closed terms in the empty definition environment of the same type σ, then $e \simeq e'$ is a formula.

The set of *provable* formulae is now defined via the following set of axioms and inference rules. We write $\vdash \varphi$ to mean "the formula φ is provable". If $n \in \mathbb{N}$, we write $\lceil n \rceil$ to mean the basic constant corresponding to n, so that $\lceil 0 \rceil = \mathsf{0}$, etc.

— $\vdash b\!\downarrow$ for each basic constant b of NRComb.
— If $\vdash e_0 e_1\!\downarrow$ then $\vdash e_0\!\downarrow$ and $\vdash e_1\!\downarrow$.
— If $\vdash e\!\downarrow$ and $\vdash e \simeq e'$ then $\vdash e'\!\downarrow$.
— If $\vdash e_1\!\downarrow$ then $\vdash \mathtt{k}\, e_0\, e_1 \simeq e_0$.
— If $\vdash e_0\!\downarrow$ and $\vdash e_1\!\downarrow$ then $\vdash \mathtt{s}\, e_0\, e_1\!\downarrow$.
— If $\vdash (e_0 e_2)(e_1 e_2)\!\downarrow$ then $\vdash \mathtt{s}\, e_0\, e_1\, e_2 \simeq (e_0 e_2)(e_1 e_2)$.
— If $\vdash e_1\!\downarrow$ then $\vdash \mathtt{fst}\,(\mathtt{pair}\, e_0\, e_1) \simeq e_0$.
— If $\vdash e_0\!\downarrow$ then $\vdash \mathtt{snd}\,(\mathtt{pair}\, e_0\, e_1) \simeq e_1$.
— $\vdash \mathtt{suc}\lceil n\rceil \simeq \lceil n+1\rceil$ for each $n \in \mathbb{N}$.
— If $\vdash e_1\!\downarrow$ then $\vdash \mathtt{rec}\, e_0\, e_1 \mathtt{0} \simeq e_0$.
— If $\vdash e_1\lceil n\rceil(\mathtt{rec}\, e_0\, e_1\lceil n\rceil)\!\downarrow$ then $\vdash \mathtt{rec}\, e_0\, e_1\lceil n+1\rceil \simeq e_1\lceil n\rceil(\mathtt{rec}\, e_0\, e_1\lceil n\rceil)$.
— If $\vdash e_0\!\downarrow$ then $\vdash \mathtt{y}\, e_0\!\downarrow$.
— If $\vdash e_0(\mathtt{y}\, e_0)e_1\!\downarrow$ then $\vdash \mathtt{y}\, e_0\, e_1 \simeq e_0(\mathtt{y}\, e_0)e_1$.
— $\vdash e \simeq e$ for any e.
— If $\vdash e \simeq e'$ then $\vdash e' \simeq e$.
— If $\vdash e \simeq e'$ and $\vdash e' \simeq e''$ then $\vdash e \simeq e''$.
— If $\vdash e_0 \simeq e_0'$ and $\vdash e_1 \simeq e_1'$ then $\vdash e_0 e_1 \simeq e_0' e_1'$.

We now define a structure NRC by taking NRC_σ to be the set of terms $e : \sigma$ such that $\vdash e\!\downarrow$, modulo the equivalence relation given by $\vdash e \simeq e'$. It is easy to check that juxtaposition of terms induces well-defined partial functions $\cdot_{\sigma\tau} : \mathsf{NRC}_{\sigma\to\tau} \times \mathsf{NRC}_\sigma \rightharpoonup \mathsf{NRC}_\tau$, and that the resulting structure NRC is an NR-TPCA. Furthermore, if A is any explicit NR-TPCA then any provable formula $e\!\downarrow$ or $e \simeq e'$ holds true under the interpretation in A given by $\ll\!-\!\gg$ (this is shown by an easy induction on proofs), so we obtain a homomorphism $h : \mathsf{NRC} \to A$ — that is, a family of total functions $h_\sigma : \mathsf{NRC}_\sigma \to A_\sigma$ such that

$$\forall f \in \mathsf{NRC}_{\sigma\to\tau},\, x \in \mathsf{NRC}_\sigma.\;\; h_{\sigma\to\tau}(f)\cdot h_\sigma(x) \succeq h_\tau(f\cdot x)$$

We may therefore think of NRC as computationally the "weakest" of all NR-TPCAs.

We say an element $a \in A_\sigma$ is NRComb-*definable* if $a = \ll e \gg$ for some closed NRComb term $e : \sigma$, and that a is *properly* NRComb-*definable* if $a = \ll e \gg$ for some term e with $\vdash e\!\downarrow$. In the latter case, the same term e is guaranteed to define an element in any NR-TPCA whatever.

We may also obtain an "infinitary" version of this model as follows: extend the definition of NRComb by adding a new basic constant $\mathtt{c}_f : \overline{1}$ for every set-theoretic function $f : \mathbb{N} \to \mathbb{N}$, and extend the above proof system with the clause:

— $\vdash \mathtt{c}_f\lceil n\rceil \simeq \lceil f(n)\rceil$ for every $f : \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$.

The effect of this is to add an oracle for every function $\mathbb{N} \to \mathbb{N}$. We refer to this extended language as "infinitary NRComb" or NRComb$^\infty$. Clearly, the corresponding term model NRC^∞ is an NR-TPCA which admits a homomorphism into any NR-TPCA A in which all functions $\mathbb{N} \to \mathbb{N}$ are representable.†

A minor variation on our construction also allows us to define a "term model for

† The existence of such a homomorphism requires the axiom of choice in general, but can be more constructively defined in the cases we shall consider; *cf.* Definition 5.5.

PCF" (again, this will feature in our conceptual discussion in Section 6.1 but is not formally required for our proofs). Suppose we extend the above proof system (in its finitary version) with the rule

— $\vdash e\downarrow$ for every e.

(this of course allows the whole system to be simplified considerably). The effect of this rule is not to say that every program terminates, but merely that non-terminating programs are concretely represented by elements of the term model. We denote the corresponding syntactic model by PCF; it may be regarded as a term model for a presentation of PCF similar to the combinatory version of (Milner 1977). Likewise, by adding the above rule to the definition of NRComb$^+$, we obtain a term model PCF^∞ for "infinitary PCF". Clearly the application operations in these structures are total.

Actually, these structures are not particularly natural ones from a mathematical point of view, as they are somewhat dependent on our choice of presentation of PCF. A much more canonical pair of structures to consider are their "observational quotients" $\mathsf{Q}^{\mathit{eff}} = \mathsf{PCF}/{\approx_{obs}}$ and $\mathsf{Q} = \mathsf{PCF}^\infty/{\approx_{obs}}$, where for $a, a' \in \mathsf{PCF}_\sigma$ we define

$$a \approx_{obs} a' \;\text{ iff }\; \forall f \in \mathsf{PCF}_{\sigma\to\overline{0}}.\, \forall n \in \mathbb{N}.\, (f \cdot a = \widehat{n} \;\Leftrightarrow\; f \cdot a' = \widehat{n})$$

All presentations of sequential, call-by-name PCF appearing in the literature give rise to this same structure $\mathsf{Q}^{\mathit{eff}}$, which we may regard as the "partial type structure of PCF-definable functionals". For a review of known results concerning Q and $\mathsf{Q}^{\mathit{eff}}$, see *e.g.* (Longley 2005a).

In the present paper, the language NRComb will play the role played by PCF in (Normann 2000). We may think of NRComb as a slightly weaker language than PCF, in that all provably defined terms of NRComb are immediately terms of PCF but not *vice versa*. Nevertheless, we will show that NRComb is sufficient to support the construction of Normann's programs and the variants of them that we will consider.

Term models for many interesting extensions of PCF may also be obtained in a similar manner by adding further operators to the definition of NRComb. See *e.g.* (Longley 2003) for a selection of such languages.

4. The extensional collapse construction

Certain special TPCAs can be regarded as consisting essentially of total functionals of finite type over the natural numbers:

Definition 4.1. (i) A *total type structure* (or TTS) is an N-TPCA A equipped with a choice of numerals $\widehat{0}, \widehat{1}, \ldots \in A_0$, such that

— every element of A_0 is a numeral $\widehat{n}$ for some $n \in \mathbb{N}$, and $\widehat{n} = \widehat{m}$ implies $n = m$;
— A is *total*, *i.e.* for any $f \in A_{\sigma\to\tau}$, $x \in A_\sigma$ we have $f \cdot x\downarrow$;
— A has *surjective pairing*, *i.e.* for any $z \in A_{\sigma\times\tau}$ we have

$$z = \mathbf{pair} \cdot (\mathbf{fst} \cdot z) \cdot (\mathbf{snd} \cdot z)$$

— A is *extensional*, *i.e.* for any $f, g \in A_{\sigma\to\tau}$ we have

$$(\forall x \in A_\sigma.\, f \cdot x = g \cdot x) \implies f = g$$

(ii) A TTS is *canonical* if $A_0 = \mathbb{N}$ (with $\widehat{n} = n$ for all n), $A_{\sigma\times\tau} = A_\sigma \times A_\tau$ with $\mathbf{pair}_{\sigma\tau} \cdot x \cdot y = (x, y)$, and $A_{\sigma\to\tau}$ is some set of functions $A_\sigma \to A_\tau$ with $f \cdot_{\sigma\tau} x = f(x)$.

Clearly, every TTS A is isomorphic to a unique canonical one (which we will denote by $\mathsf{Can}(A)$), and the isomorphism $A \cong \mathsf{Can}(A)$ is itself unique. We will write β^A for the isomorphism $A \to \mathsf{Can}(A)$; however, to simplify notation we will often blur the distinction between A and $\mathsf{Can}(A)$ when there is no danger of confusion.

It is easy to see that no TTS can ever be an NR-TPCA, since one can construct elements of type $\overline{1} \to \overline{1}$ with no pointwise fixed point.

Any NR-TPCA A gives rise to a total type structure by means of the following standard construction. Recall that a *partial equivalence relation* or *PER* on a set X is just a symmetric, transitive relation on X (that is, an equivalence relation on a subset of X). If $\sim$ is a PER on X, we write $X/\!\sim$ for the set of equivalence classes for $\sim$, and $[x]^\sim$ (or $[x]$ if $\sim$ is evident) for the equivalence class (if there is one) containing a particular $x \in X$.

Definition 4.2 (Extensional collapse). Let A be any TPCA, and $\sim$ any PER on A_0.

(i) Define partial equivalence relations $\sim_\sigma$ on the sets A_σ as follows:

— $\sim_{\overline{0}} \;=\; \sim$
— $z \sim_{\sigma\times\tau} w$ iff $\mathbf{fst} \cdot z \sim_\sigma \mathbf{fst} \cdot w$ and $\mathbf{snd} \cdot z \sim_\tau \mathbf{snd} \cdot w$.
— $f \sim_{\sigma\to\tau} g$ iff for all $x, y \in A_\sigma$, $x \sim_\sigma y$ implies $f \cdot x \sim_\tau g \cdot y$.

The family of PERs $\sim_\sigma$ is called the *logical relation* on A *induced by* $\sim$.

(ii) We now define $\mathsf{EC}(A, \sim)$, the *extensional collapse* of A with respect to $\sim$, to be the family of sets $\mathsf{EC}(A)_\sigma = A_\sigma/\!\sim_\sigma$, with application operations $\cdot_{\sigma\tau}$ defined by $[f]\cdot[x] = [f\cdot x]$.

(iii) If moreover A is an N-TPCA and $\sim$ is defined by $x \sim y$ iff $x = y = \widehat{n}$ for some n, the family of PERs $\sim_\sigma$ is called the *standard logical relation* on A, written as $\sim^A$, and $\mathsf{EC}(A, \sim)$ is called the *standard extensional collapse* of A and denoted simply by $\mathsf{EC}(A)$. In this case, we also write $\mathsf{Tot}(A_\sigma)$ for the set $\{x \in A_\sigma \mid x \sim_\sigma x\}$ of *total elements* of A.

It is routine to verify the following:

Proposition 4.3. For any non-trivial N-TPCA A, $\mathsf{EC}(A)$ is a total type structure.

We may therefore define the *canonical extensional collapse* $\mathsf{EC}^{\mathsf{C}}(A)$ of A to be the total type structure $\mathsf{Can}(\mathsf{EC}(A))$.

A categorical perspective on these ideas will also be useful. The following definition (in essence) appears in (Longley 1999b; Lietz and Streicher 2002):

Definition 4.4. Let A be any TPCA. The category $\mathbf{Asm}(A)$ of *assemblies* over A is defined as follows:

— An object X is a triple $(|X|, \rho_X, \Vdash_X)$ where $|X|$ is a set, ρ_X is a type, and $\Vdash_X \subseteq A_{\rho_X} \times |X|$ is a relation such that for each $x \in |X|$ there is some $a \in A_{\rho_X}$ with $a \Vdash_X x$.
— A morphism $f : X \to Y$ is a function $f : |X| \to |Y|$ that is *tracked* by some $t \in A_{\rho_X\to\rho_Y}$, in the sense that for any $x \in |X|$ and $a \in A_{\rho_X}$, if $a \Vdash_X x$ then $t \cdot a \Vdash_Y f(x)$.

An assembly X is a *modest set* if we have $a \Vdash_X x$ and $a \Vdash_X x'$ only when $x = x'$. We write $\mathbf{Mod}(A)$ for the full subcategory of $\mathbf{Asm}(A)$ consisting of modest sets.

If X is an assembly, we will often write $x \in X$ to mean $x \in |X|$, and write "a realizes $x \in X$" to mean $a \Vdash_X x$.

Any modest set X gives rise to a PER $\sim_X$ on A_{ρ_X}: take $a \sim_X b$ iff a, b both realize the same $x \in X$. Conversely, any PER $\sim$ on any A_σ may be identified with the modest set $(A_\sigma/\sim, \sigma, \in)$.

The following facts are routinely checked. The proofs are straightforward adaptations of the proofs for the untyped case given *e.g.* in (Longley 1995); see also (Lietz and Streicher 2002).

Proposition 4.5. Let A be any TPCA.

(i) The category $\mathbf{Asm}(A)$ is well-pointed (that is, the functor $\mathrm{Hom}(1, -)$ is faithful) — indeed, for any object X we have $\mathrm{Hom}(1, X) \cong |X|$.

(ii) $\mathbf{Asm}(A)$ is cartesian closed — indeed, for any objects X, Y we may define Y^X by

$$|Y^X| \;=\; \mathrm{Hom}(X, Y), \qquad \rho_{Y^X} \;=\; \rho_X \to \rho_Y, \qquad t \vdash_{Y^X} f \;\text{ iff }\; t \text{ tracks } f : X \to Y$$

Moreover, the subcategory $\mathbf{Mod}(A)$ is closed under exponentiation.

(iii) If A is an N-TPCA, $\mathbf{Asm}(A)$ has a natural number object N given by: $|N| = \mathbb{N}$, $\rho_N = \overline{0}$, $a \Vdash_N n$ iff $a = \widehat{n}$. This object is modest if A is non-trivial.

In general, let us say a category $\mathcal{C}$ has a *standard natural number object* if it has a natural number object N together with a canonical bijection $\mathrm{Hom}(1, N) \cong \mathbb{N}$. Suppose $\mathcal{C}$ has finite products and a standard natural number object N. By an *internal total type structure* $\mathbf{T}$ in $\mathcal{C}$ we mean a family of objects $\mathbf{T}_\sigma$ in $\mathcal{C}$ together with an isomorphism $\mathbf{T}_0 \cong N$ and "application" morphisms $\mathbf{T}_{\sigma\to\tau} \times \mathbf{T}_\sigma \to \mathbf{T}_\tau$, such that the sets $\mathrm{Hom}(1, \mathbf{T}_\sigma)$ constitute a TTS denoted by $\mathrm{Hom}(1, \mathbf{T})$ (with the canonical choice of numerals, and with application functions induced by the application morphisms). If $\mathcal{C}$ is also well-pointed and cartesian closed, there is a standard choice of an internal TTS in $\mathcal{C}$, given by

$$\mathbf{T}_0 = N, \qquad \mathbf{T}_{\sigma\to\tau} = \mathbf{T}_\tau^{\mathbf{T}_\sigma}$$

with application given by the evaluation morphisms. In the case $\mathcal{C} = \mathbf{Asm}(A)$, we write $\mathbf{EC}(A)$ for this internal TTS; we may also write $|\mathbf{EC}| \cong \mathrm{Hom}(1, \mathbf{EC})$ for the evident total type structure given by $|\mathbf{EC}|_\sigma = |\mathbf{EC}_\sigma|$. It is easy to see that $|\mathbf{EC}(A))| = \mathsf{EC}^{\mathsf{C}}(A)$; indeed, the object $\mathbf{EC}(A)_\sigma$ may be explicitly described as follows:

$$|\mathbf{EC}(A)_\sigma| \;=\; \mathsf{EC}^{\mathsf{C}}(A)_\sigma, \qquad \rho_{\mathbf{EC}(A)_\sigma} \;=\; \sigma, \qquad a \Vdash_{\mathbf{EC}(A)_\sigma} \beta^{\mathsf{EC}(A)}([x]) \text{ iff } a \in [x]$$

If A is non-trivial then $\mathbf{EC}(A)$ lies entirely within $\mathbf{Mod}(A)$.

Since $\mathbf{Asm}(A)$ is often called a "standard realizability model" over A, we may think of $\mathsf{EC}(-)$ as the *standard realizability construction* of a TTS over a given N-TPCA.

In general, knowing that two TPCAs are related in some way does not imply any particular relationship between their extensional collapses, except in rather trivial cases. For instance, it may happen that A is a sub-TPCA of B and $\sim$ is a PER on A_0, but that $\mathsf{EC}(A, \sim)$ is smaller than, or incomparable with, $\mathsf{EC}(B, \sim)$. Even if A is a submodel of B, B a submodel of C and $\mathsf{EC}(A, \sim) \cong \mathsf{EC}(C, \sim)$, it may happen that $\mathsf{EC}(B, \sim)$ is larger

than, smaller than or incomparable with $\mathsf{EC}(A, \sim)$.[‡] Moreover, even when two extensional collapses do coincide, it is frequently non-trivial to prove this. All these phenomena stem from the fact that the construction $\mathsf{EC}(-)$ is highly *non-functorial* (that is, it does not interact well with morphisms between TPCAs in the sense of (Longley 1999b; Lietz and Streicher 2002)). The root of this problem is of course the fact that the relation $\sim_{\sigma\to\tau}$ depends contravariantly on $\sim_\sigma$.

Seen in this light, the fact that several known models give rise the same extensional collapse would not in itself seem to provide particularly strong evidence that *all* TPCAs in some natural class would do so. Indeed, the fact that results to this effect can sometimes be obtained came as a considerable surprise to the author, and it is this possibility of establishing the extensional collapse of a whole class of models at once which constitutes the main contribution of this paper.

We now comment briefly on two apparent generalizations of our definitions.

Remark 4.6. (i) A superficially more general definition of an N-TPCA might allow a non-empty *set* of encodings for each natural number k rather than just a single element $\widehat{k}$. In other words, in place of the $\widehat{k}$ we could take a PER $\approx_0$ on any A_0 together with a bijection $A_0/{\approx_0} \cong \mathbb{N}$, such that the successor and recursor operations were realizable in A in a suitable sense. This would then induce PERs $\approx_\sigma$ at higher types as above; we would obtain a type structure $\mathsf{EC}(A, \approx)$ and a corresponding internal type structure $\mathbf{EC}(A, \approx)$ in $\mathbf{Asm}(A)$.

However, any such PER $\approx_0$, considered as a modest set, is in fact isomorphic in $\mathbf{Asm}(A)$ to a PER $\sim_0$ arising from a choice of numerals $\widehat{n}$ as in Definition 4.2. Specifically, for $\widehat{0}$ take any realizer for 0 in A_0, and inductively define $\widehat{n+1} = \mathbf{suc}' \cdot \widehat{n}$, where $\mathbf{suc}' \in A_1$ is a realizer for the successor operation with respect to $\approx_0$. Now let $x \sim_0 y$ iff $x = y = \widehat{n}$ as above. Clearly, each $\widehat{n}$ is a realizer for n with respect to $\approx_0$, so the obvious mapping $A_0/{\sim_0} \to A_0/{\approx_0}$ is realized by the identity combinator. Conversely, if $\mathbf{rec}'$ realizes the appropriate recursor operation with respect to $\approx_0$ and z is any $\approx_0$-realizer for n then

$$\mathbf{rec}' \cdot \widehat{0} \cdot (\mathbf{s} \cdot \mathbf{k} \cdot (\mathbf{k} \cdot \mathbf{suc}')) \cdot z \;=\; \widehat{n}$$

so we have a realizer for the inverse mapping. Thus $\approx_0 \cong \sim_0$ in $\mathbf{Asm}(A)$.

It follows easily that $\approx_\sigma \cong \sim_\sigma$ at all types σ, and hence that $\mathsf{EC}(A, \approx)$ coincides with $\mathsf{EC}(A)$ as given by Definition 4.2.

(ii) Another, even more superficial, generalization could be obtained by allowing the natural numbers to be represented by a PER $\approx$ at a type υ other than $\overline{0}$. Such a PER would not necessarily be isomorphic to any PER on $\overline{0}$. However, we may define a type $\sigma[\upsilon]$ for each type σ as follows:

$$\overline{0}[\upsilon] \;=\; \upsilon, \qquad (\sigma \times \tau)[\upsilon] \;=\; \sigma[\upsilon] \times \tau[\upsilon], \qquad (\sigma \to \tau)[\upsilon] \;=\; \sigma[\upsilon] \to \tau[\upsilon]$$

[‡] Many examples of this kind may be obtained by considering term models for the languages considered in (Longley 1999a) and mild variations on them, taking $\sim$ to be the evident PER corresponding to $N_\perp$. The inclusions between these languages illustrate a variety of pathological possibilities; details may appear elsewhere.

We may then define an N-TPCA $A[\upsilon]$ just by setting $A[\upsilon]_\sigma = A_{\sigma[\upsilon]}$, and $\approx$ would then give a natural number object in $\mathbf{Asm}(A[\upsilon])$ as above.

Thus, neither of these generalizations would enlarge the class of type structures that we are able to obtain via the extensional collapse construction.

4.1. *A relative version*

In the definition of TPCAs above, operations of higher type are treated as entities of the same kind as the data they act on: both are simply elements of A. Thus, TPCAs are a suitable framework for notions such as "continuous operations acting on continuous data", or "effective operations acting on effective data". However, some refinements to this framework are needed if we wish to consider hybrid notions such as "effective operations acting on continuous data" — or, more generally, a notion of operations of some restricted class acting on data in some wider class. We may capture such notions by means of a *relative* version of the extensional collapse construction, which we now introduce.

Definition 4.7 (Relative extensional collapse). (i) A *substructure* T' of a type structure T is simply a sub-N-TPCA of T (with respect to the standard choice of numerals in T).

(ii) If A' is a sub-N-TPCA of A, we define $\mathsf{EC}(A; A')$, the *relative extensional collapse* of A and A', to be the substructure of $\mathsf{EC}(A)$ consisting of those elements t that are realized by at least one element of A'.

It is easy to check that $\mathsf{EC}(A; A')$, so defined, is indeed a substructure of $\mathsf{EC}(A)$. We also write $\mathsf{EC}^{\mathsf{C}}(A; A')$ for the substructure of $\mathsf{EC}^{\mathsf{C}}(A)$ corresponding to $\mathsf{EC}(A; A')$. Again, these ideas have a natural categorical formulation:

Definition 4.8. Let A be any TPCA, A' a sub-TPCA of A. The category $\mathbf{Asm}(A; A')$ is defined as follows: objects are objects of $\mathbf{Asm}(A)$, and morphisms are morphisms of $\mathbf{Asm}(A)$ that are tracked by some $t \in A'$.

The category $\mathbf{Asm}(A; A')$ is cartesian closed, and in fact has precisely the same exponentials as $\mathbf{Asm}(A)$. If A' is a sub-N-TPCA of A then $\mathbf{Asm}(A; A')$ has an evident natural number object N, and as before we may define a family of objects $\mathbf{EC}(A; A')_\sigma$ by

$$\mathbf{EC}(A; A')_0 \;=\; N, \qquad \mathbf{EC}(A; A')_{\sigma\to\tau} \;=\; \mathbf{EC}(A; A')_\tau^{\mathbf{EC}(A;A')_\sigma}$$

It is easy to see that $\mathrm{Hom}(1, \mathbf{EC}(A; A')_\sigma)$ may be canonically identified with $\mathsf{EC}(A; A')_\sigma$. The category $\mathbf{Asm}(A; A')$ is often called the *relative realizability* model over A with respect to A' (see *e.g.* (Awodey *et al.* 2002)). As a trivial point, note that $\mathbf{EC}(A; A) = \mathbf{EC}(A)$.

Note also that $\mathsf{EC}(A; A')$ is itself an N-TPCA, but need not be a type structure in its own right since it might not be extensional. That is, there might be two elements $f, g \in \mathsf{EC}(A; A')_{\sigma\to\tau}$ which are distinct as functions $\mathsf{EC}(A)_\sigma \to \mathsf{EC}(A)_\tau$, but which restrict to the same function $\mathsf{EC}(A; A')_\sigma \to \mathsf{EC}(A; A')_\tau$. However, in the main examples of interest

in this paper, $\mathsf{EC}(A; A')$ will be the substructure $\mathsf{C}^{\mathit{eff}}$ of C. As we shall shortly see, this structure is in fact extensional, and so can be considered as a type structure in itself.

4.2. *Known constructions of total type structures*

We next review some known results concerning the extensional collapse of some of the models considered in Section 3. These will give us several possible definitions of the three type structures that are our primary objects of interest.

The type structure C of *continuous functionals* is a canonical TTS which may be specified in many equivalent ways. The following two characterizations will play a central role in this paper:

— $\mathsf{C} \cong \mathsf{EC}(\mathsf{P})$. This characterization appeared in (Ershov 1972b; Ershov 1974), and can be seen as a cleaned-up version of Kreisel's original definition in (Kreisel 1959). This is the construction that is usually favoured as the definition of C in the more recent literature (see *e.g.* (Normann 1999)); here we shall adopt it as our official definition of C. Since all the relevant domains P_σ are known to be retracts of Plotkin's $\mathbb{T}^\omega$ (Plotkin 1978), it follows easily from this characterization that also $\mathsf{C} \cong \mathsf{EC}(\mathbb{T}^\omega)$.

— $\mathsf{C} \cong \mathsf{EC}(K_2)$. This is essentially Kleene's original definition of C via *associates* (Kleene 1959b). Modulo unimportant details, an associate for a functional is essentially a realizer in K_2. The equivalence between this and the P characterization is non-trivial and was first explicitly proved in (Hinata and Tugué 1969); see also (Hyland 1975; Bauer 2002). An alternative proof of the equivalence will be given under Theorem 5.10 below.

The following characterizations are not technically required for this paper, but reinforce the impression that C is a mathematically natural structure.

— $\mathsf{C} \cong \mathsf{EC}(\mathsf{L})$, where L is the TPCA arising from the category of *continuous lattices* (taking $\bar{N}$ to be the "flat" lattice of natural numbers with a bottom and top element). This characterization is due to Scott (Scott 1976), and is quite close to Ershov's characterization via P. Since all countably-based continuous lattices are known to be retracts of Scott's $\mathcal{P}\omega$ (Scott 1976), it follows easily that $\mathsf{C} \cong \mathsf{EC}(\mathcal{P}\omega)$.

— In (Bergstra 1978), C is characterized as the maximal TTS closed under Kleene's S1–S9 computability and not containing ${}^2E$, the existential quantifier for predicates on the natural numbers.

— In (Normann *et al.* 1999), C is characterized as a certain extensional collapse of the product over n of the full set-theoretic type structures over $\{0, \ldots, n\}$. This is an example of an extensional collapse construction using an N-TPCA that is not an NR-TPCA.

— Most of the other known constructions of C are of a topological or semi-topological character: for instance, one obtains C as the type structure over $\mathbb{N}$ in the cartesian closed category of compactly generated Hausdorff spaces, or of filter spaces, limit spaces, etc. (see *e.g.* (Hyland 1979; Escardó *et al.* 2004)). These characterizations do not fit into the realizability framework considered here.

Many of the above constructions can be effectivized in a natural way, and hence allow

us to define an "effective" substructure of C. In fact, all reasonable effectivizations seem to lead to the same effective substructure $\mathsf{C}^{\mathit{eff}} \subset \mathsf{C}$ of *effective continuous functionals*. Thus:

— $\mathsf{C}^{\mathit{eff}} \cong \mathsf{EC}(\mathsf{P}; \mathsf{P}^{\mathit{eff}})$ (Ershov 1972b). We adopt this as our official definition of $\mathsf{C}^{\mathit{eff}}$.
— $\mathsf{C}^{\mathit{eff}} \cong \mathsf{EC}(K_2; K_2^{\mathit{eff}})$ (Kleene 1959b).
— $\mathsf{C}^{\mathit{eff}} \cong \mathsf{EC}(\mathsf{L}; \mathsf{L}^{\mathit{eff}}) \cong \mathsf{EC}(\mathcal{P}\omega; \mathcal{P}\omega^{\mathit{eff}})$ (Scott 1976; Beeson 1985).
— Some of the topological models, such as the category of filter spaces, have a natural effective submodel, and these too give rise to $\mathsf{C}^{\mathit{eff}}$ (Hyland 1979).

Now suppose that for each of these models we consider the effective submodel as a TPCA in its own right, so that we are only considering "effective operations acting on effective data". In this case, the extensional collapse construction yields a third class of functionals: the type structure $\mathsf{C}^{\mathit{heff}}$ of the so-called *hereditarily effective continuous functionals*. Here, a purely topological characterization is not available, but instead we have an additional realizability characterization.

— $\mathsf{C}^{\mathit{heff}} \cong \mathsf{EC}(\mathsf{P}^{\mathit{eff}})$ (Ershov 1976a). We adopt this as our official definition of $\mathsf{C}^{\mathit{heff}}$.
— Most remarkably, $\mathsf{C}^{\mathit{heff}}$ coincides with the type structure HEO of *hereditarily effective operations*, defined as $\mathsf{EC}(K_1)$. This non-trivial result is known as the *generalized Kreisel-Lacombe-Shoenfield (KLS) theorem*. The ordinary KLS theorem, first proved in (Kreisel, Lacombe and Shoenfield 1959), essentially says that $\mathsf{C}_2^{\mathit{heff}} = \mathsf{HEO}_2$. The generalization to higher types was noted in (Kreisel 1959); the first complete proofs appeared in (Hyland 1975; Ershov 1976a). The treatment in (Berger 1993) is particularly well adapted to our present purposes.
— $\mathsf{C}^{\mathit{heff}} \cong \mathsf{EC}(K_2^{\mathit{eff}})$. A proof of the equivalence $\mathsf{HEO} \cong \mathsf{EC}(K_2^{\mathit{eff}})$ appears in (Troelstra 1973, §2.6).
— $\mathsf{C}^{\mathit{heff}} \cong \mathsf{EC}(\mathsf{L}^{\mathit{eff}}) \cong \mathsf{EC}(\mathcal{P}\omega^{\mathit{eff}})$ (Beeson 1985).

For our purposes, the characterizations via K_1 and via $\mathsf{P}^{\mathit{eff}}$ will be the most important ones. We will henceforth tend to use the name HEO for this type structure, except in contexts that pertain specifically to the characterization via $\mathsf{P}^{\mathit{eff}}$.

As noted in the Introduction, HEO is "incomparable" with $\mathsf{C}^{\mathit{eff}}$ and indeed with C. For instance, the *Kleene tree functional* is present in HEO_2 but has no counterpart in $\mathsf{C}^{\mathit{eff}}$ or C, whilst the *fan functional* in $\mathsf{C}_3^{\mathit{eff}}$ has no counterpart in HEO. These facts will not be required for this paper, but we refer the interested reader to (Longley 2005a) for further information.

The above is not an exhaustive list of the known characterizations of our type structures, and we shall have occasion to mention a few others below. The characterizations via P and $\mathsf{P}^{\mathit{eff}}$ are of particular significance, since the finite elements of P give rise to "neighbourhood systems" on our total type structures which will play an important role. For any $\mathfrak{c} \in \mathsf{P}_\sigma^{\mathit{comp}}$, let us define

$$\begin{array}{rcll} U_{\mathfrak{c}}^{\mathsf{C}} & = & \{\beta([\mathfrak{u}]) \mid \mathfrak{u} \in \mathsf{Tot}(\mathsf{P}_\sigma), \mathfrak{c} \sqsubseteq \mathfrak{u}\} & \subseteq \quad \mathsf{C}_\sigma \\ U_{\mathfrak{c}}^{\mathsf{RC}} & = & \{\beta([\mathfrak{u}]) \mid \mathfrak{u} \in \mathsf{Tot}(\mathsf{P}_\sigma) \cap \mathsf{P}_\sigma^{\mathit{eff}}, \mathfrak{c} \sqsubseteq \mathfrak{u}\} & \subseteq \quad \mathsf{C}_\sigma^{\mathit{eff}} \subset \mathsf{C}_\sigma \\ U_{\mathfrak{c}}^{\mathsf{HRC}} & = & \{\beta([\mathfrak{u}]) \mid \mathfrak{u} \in \mathsf{Tot}(\mathsf{P}_\sigma^{\mathit{eff}}), \mathfrak{c} \sqsubseteq \mathfrak{u}\} & \subseteq \quad \mathsf{C}_\sigma^{\mathit{heff}} \end{array}$$

We write simply $U_{\mathfrak{c}}$ when it is clear which type structure we are talking about. If T is one of $\mathsf{C}, \mathsf{C}^{\mathit{eff}}, \mathsf{C}^{\mathit{heff}}$, we may refer to the sets $U_{\mathfrak{c}}^T$ as *Ershov neighbourhoods* within T.

An important fact about our three type structures is the Kleene-Kreisel *density theorem* (Kleene 1959b; Kreisel 1959). In its simplest form, this states that all of the above neighbourhoods are in fact inhabited. For our purposes we will require a stronger "effective" form of the theorem, similar to the one appearing in (Kleene 1959b, §2.1). Since our version of this theorem is slightly idiosyncratic in some respects, we include a proof, which is modelled on the concise argument given in (Normann 1999).

Theorem 4.9 (Effective density theorem). Let σ be any type. Then for each code $c \in \mathbb{N}$ of type σ, there exist an element $\mathfrak{x}_c \in \mathsf{Tot}(\mathsf{P}_\sigma) \cap \mathsf{Tot}(\mathsf{P}_\sigma^{\mathit{eff}})$ and an associated total function $\xi_{\sigma,c} : \mathbb{N} \to \mathbb{N}$, computable uniformly in c, such that

(1) each $\xi_{\sigma,c}(i)$ is a basic ζ_σ code, and $\xi_{\sigma,c}$ enumerates the basic compacts below $\mathfrak{x}_c$ (*i.e.*, $\{\zeta_\sigma(\xi_{\sigma,c}(i)) \mid i \in \mathbb{N}\} = B_{\mathfrak{x}_c}$),
(2) if $c \in \operatorname{dom} \zeta_\sigma$ then $\zeta_\sigma(c) \sqsubseteq \mathfrak{x}_c$,
(3) each basic compact below $\mathfrak{x}_c$ appears infinitely many times in $\xi_{\sigma,c}$.

Thus, if $\mathfrak{c} = \zeta_\sigma(c)$ then $\mathfrak{x}_c$ represents an element in each of $U_{\mathfrak{c}}^{\mathsf{C}}, U_{\mathfrak{c}}^{\mathsf{RC}}, U_{\mathfrak{c}}^{\mathsf{HRC}}$.

Proof. We will prove the theorem for the pure types (which is actually all that we shall require); the adaptation to arbitrary types is then straightforward. We first establish the theorem without condition (3).

If $\sigma = \overline{0}$ or $\sigma = \overline{1}$ then the result is trivial, so suppose $\sigma = \overline{k}$ where the result holds for type $\overline{k-2}$. Take $c \in \mathbb{N}$; we will construct a suitable element $\mathfrak{x}_c$ together with an enumeration $\xi_{\sigma,c}$ (computable uniformly in c) of the basic compacts below $\mathfrak{x}_c$.

If $c \notin \operatorname{dom} \zeta_\sigma$, we may take $\mathfrak{x}_c$ to be the constant zero function, and $\xi_{\sigma,c}$ to be an effective enumeration of all codes $\langle b \mapsto \breve{0} \rangle$ where $b \in \operatorname{dom} \zeta_{k-1}$. Otherwise, we may write c as $\langle b_0 \mapsto \breve{n}_0, \ldots, b_{r-1} \mapsto \breve{n}_{r-1} \rangle$, where $b_i = \langle a_{i0} \mapsto \breve{m}_{i0}, \ldots, a_{i(s_i-1)} \mapsto \breve{m}_{i(s_i-1)} \rangle \in \operatorname{dom} \zeta_{k-1}$ and each $a_{ij} \in \operatorname{dom} \zeta_{k-2}$. Let $R = \{((i,j),(i',j')) \mid \zeta_{k-2}(a_{ij}),\ \zeta_{k-2}(a_{i'j'}) \text{ are consistent}\}$. Note that R is a finite set which may be effectively computed from c. For each $r = ((i,j),(i',j')) \in R$, let d_r be a code for $\zeta_{k-1}(a_{ij}) \sqcup \zeta_{k-1}(a_{i'j'})$ computed by some standard procedure; then by the induction hypothesis we have elements $\mathfrak{x}_{d_r} \in \mathsf{Tot}(\mathsf{P}_{k-2}) \cap \mathsf{Tot}(\mathsf{P}_{k-2}^{\mathit{eff}})$ and enumerations ξ_{k-2,d_r} computable uniformly in r and satisfying conditions (1) and (2) of the theorem.

Now let $\mathsf{T}_c = \{\mathfrak{G} \in \mathsf{P}_{k-1} \mid \mathfrak{G}(\mathfrak{x}_{d_r}) \in \mathbb{N} \text{ for all } r \in R\}$; note that both $\mathsf{Tot}(\mathsf{P}_{k-1}) \subseteq \mathsf{T}_c$ and $\mathsf{Tot}(\mathsf{P}_{k-1}^{\mathit{eff}}) \subseteq \mathsf{T}_c$. Let us say that $\mathfrak{G} \in \mathsf{T}_c$ is *semiconsistent* with (i, b_i) if for all $i' < r$ and $((i,j),(i',j')) \in R$ we have that $\mathfrak{G}(\mathfrak{x}_r) = \zeta_{k-1}(b_i)(\mathfrak{x}_{d_r})$. Note here that $\zeta_{k-1}(b_i)(\mathfrak{x}_{d_r}) \in \mathbb{N}$ since $\zeta_{k-2}(a_{ij}) \sqsubseteq \mathfrak{x}_{d_r}$; hence there is a continuous function $\mathfrak{T}_{c,i} : \mathsf{P}_{k-1} \to \mathsf{P}_0$ such that

$$\mathfrak{T}_{c,i}(\mathfrak{G}) \;=\; \begin{cases} 0 & \text{if } \mathfrak{G} \in \mathsf{T}_c \text{ is semiconsistent with } (i, b_i), \\ 1 & \text{if } \mathfrak{G} \in \mathsf{T}_c \text{ is not semiconsistent with } (i, b_i), \\ \bot & \text{if } \mathfrak{G} \notin \mathsf{T}_c \end{cases}$$

Moreover, an enumeration of $\mathfrak{T}_{c,i}$ may be constructed effectively in c and i.

Next, if $\mathfrak{G} \in \mathsf{T}_C$ is semiconsistent with both (i, b_i) and $(i', b_{i'})$, then $\zeta_{k-1}(b_i)$ and $\zeta_{k-1}(b_{i'})$ will themselves be consistent, since given any j, j' such that $\zeta_{k-2}(a_{ij}), \zeta_{k-2}(a_{i'j'})$ are consistent, we must have $m_{ij} = \mathfrak{G}(\mathfrak{x}_{d_r}) = m_{i'j'}$ where $r = ((i,j),(i',j'))$. Moreover, if $\zeta_{k-1}(b_i) \sqsubseteq \mathfrak{G} \in \mathsf{T}_c$ then $\mathfrak{T}_{c,i}(\mathfrak{G}) = 0$.

We may now define $\mathfrak{x}_c$ by

$$\mathfrak{x}_c(\mathfrak{G}) \;=\; \begin{cases} n_i & \text{if } \zeta_{k-1}(b_i) \sqsubseteq \mathfrak{G} \text{ or } \mathfrak{T}_{c,i}(\mathfrak{G}) = 0 \text{ for some } i, \\ 0 & \text{if } \mathfrak{T}_{c,i}(\mathfrak{G}) = 1 \text{ for all } i, \\ \bot & \text{otherwise} \end{cases}$$

Clearly $\zeta_\sigma(c) \sqsubseteq \mathfrak{x}_c$, and $\mathfrak{x}_c \in \mathsf{Tot}(\mathsf{P}_\sigma) \cap \mathsf{Tot}(\mathsf{P}_\sigma^{\mathit{eff}})$ since $\mathsf{Tot}(\mathsf{P}_{k-1}) \cup \mathsf{Tot}(\mathsf{P}_{k-1}) \subseteq \mathsf{T}_c$. Moreover, an enumeration $\xi_{\sigma,c}$ of basic codes for the basic compacts below $\mathfrak{x}_c$ may be given effectively in c.

Finally, given a family of elements $\mathfrak{x}_c$ and enumerations $\xi_{\sigma,c}$ satisfying all the required properties except condition (3), we may obtain an enumeration $\xi'_{\sigma,c}$ for $\mathfrak{x}_c$ satisfying all required conditions simply by defining $\xi'_{\sigma,c}(i) = \xi_{\sigma,c}(\mathrm{proj}\,(0,i))$. □

Condition (3) is not a standard part of the theorem, but is a minor technical convenience for our purposes, as is the fact that $\xi_{\sigma,c}$ enumerates a total element even when $c \notin \mathrm{dom}\ \zeta_\sigma$. The purpose of condition (3) is to ensure that for any t, the supremum $\bigsqcup_{i \geq t} \zeta_\sigma(\xi_{\sigma,c}(i))$ is still $\mathfrak{x}_c$.

An easy consequence of the density theorem is that if $f, g \in \mathsf{C}_{\sigma\to\tau}$ and $f \cdot x = g \cdot x$ for all $x \in \mathsf{C}_\sigma^{\mathit{eff}}$, then $f = g$. This means that $\mathsf{C}^{\mathit{eff}}$ is itself an extensional structure, so as noted at the end of Section 4.1, it can be seen as a TTS in its own right.

Clearly, any code c for type σ determines an Ershov neighbourhood $U^T_{\zeta_\sigma(c)}$ in T_σ. However, there is another way in which a code may be seen as defining a subset of T_σ without any reference to P. Suppose T is either C or $\mathsf{C}^{\mathit{heff}}$; then for any σ and any code c for type σ, we may define a set $O^T_{\sigma,c} \subseteq T_\sigma$ as follows. Let $O^T_{0,\check{\bot}} = \mathbb{N}$, $O^T_{0,\check{n}} = \{n\}$; if $c = \langle a, b\rangle$ then let $O^T_{\sigma\times\tau,c} = O^T_{\sigma,a} \times O^T_{\tau,b}$; and if $c = \langle a_0 \mapsto b_0, \ldots, a_{r-1} \mapsto b_{r-1}\rangle$ then let

$$O^T_{\sigma\to\tau,c} \;=\; \{f \in \mathsf{C}_{\sigma\to\tau} \mid \forall i < r.\, \forall x \in T_\sigma.\ x \in O^T_{\sigma,a_i} \;\Rightarrow\; f(x) \in O^T_{\tau,b_i}\}$$

We may also define $O^{\mathsf{RC}}_{\sigma,c} = O^{\mathsf{C}}_{\sigma,c} \cap \mathsf{C}^{\mathit{eff}}_\sigma$. We refer to the sets $O^T_{\sigma,c}$ as *Kreisel neighbourhoods* within T;[§] again, we write simply $O_{\sigma,c}$ if it is clear what T is, or even O_c if σ is also apparent.

A pleasing fact is that the Ershov and Kreisel notions of neighbourhood coincide. This result is surely folklore, though we have been unable to find an explicit statement of it in the literature.[¶]

Proposition 4.10. Let T be one of $\mathsf{C}, \mathsf{C}^{\mathit{eff}}, \mathsf{C}^{\mathit{heff}}$, and σ any type.
(i) For any $\mathfrak{c}, \mathfrak{c}' \in \mathsf{P}^{\mathit{comp}}_\sigma$, $U^T_{\mathfrak{c}} \cap U^T_{\mathfrak{c}'}$ is inhabited iff $\mathfrak{c}, \mathfrak{c}'$ are consistent.
(ii) If $\mathfrak{c} = \zeta_\sigma(c)$ then $U^T_{\mathfrak{c}} = O^T_{\sigma,c}$.

Proof. Let A denote P if T is C or $\mathsf{C}^{\mathit{eff}}$, and $\mathsf{P}^{\mathit{eff}}$ if T is $\mathsf{C}^{\mathit{heff}}$. In addition, let A' denote

[§] Kreisel's definition in (Kreisel 1959) did not feature a neighbourhood corresponding to $\check{\bot}$; moreover, Kreisel's approach was to construct the elements of C from the neighbourhood structure rather than *vice versa*. The only justification for our terminology is that Kreisel's neighbourhoods were considered as properties of total objects without reference to the partial ones.

[¶] Some closely related facts appear in (Hyland 1975, §1.3), though the proof given there only applies to the case $T = \mathsf{C}$.

P if T is C, and $\mathsf{P}^{\mathit{eff}}$ if T is $\mathsf{C}^{\mathit{eff}}$ or $\mathsf{C}^{\mathit{heff}}$. Let $W_\sigma = \mathsf{Tot}(A_\sigma) \cap A'_\sigma$; we will say $\mathfrak{x} \in W_\sigma$ *represents* $x \in T_\sigma$ if $x = \beta^{\mathsf{EC}(A)}([\mathfrak{x}])$.

(i) If $\mathfrak{c}, \mathfrak{c}'$ are consistent then $\mathfrak{c} \sqcup \mathfrak{c}'$ exists and is $\zeta_\sigma(c'')$ for some c''. Take $\mathfrak{x}_{c''} \in W_\sigma$ with $\mathfrak{x}_{c''} \sqsupseteq \mathfrak{c} \sqcup \mathfrak{c}'$ as in Theorem 4.9; then clearly $\beta([\mathfrak{x}_{c''}]) \in U_\mathfrak{c} \cap U_{\mathfrak{c}'}$.

Conversely, suppose $f \in U_\mathfrak{c} \cap U_{\mathfrak{c}'}$, and take $\mathfrak{f} \sqsupseteq \mathfrak{c}$, $\mathfrak{f}' \sqsupseteq \mathfrak{c}'$ both representing f. To show that $\mathfrak{c}, \mathfrak{c}'$ are consistent, it suffices to show that if $\mathfrak{b} \sqsubseteq \mathfrak{c}$ and $\mathfrak{b}' \sqsubseteq \mathfrak{c}'$ are basic compacts then $\mathfrak{b}, \mathfrak{b}'$ are consistent. Write σ as $\sigma_0 \to \cdots \to \sigma_{n-1} \to \overline{0}$; then any such $\mathfrak{b}$ has the form $\mathfrak{a}_0 \Rightarrow \cdots \Rightarrow \mathfrak{a}_{n-1} \Rightarrow n$ where $n \in \mathbb{N}$; likewise any such $\mathfrak{b}'$ has the form $\mathfrak{a}'_0 \Rightarrow \cdots \Rightarrow \mathfrak{a}'_{n-1} \Rightarrow n'$. Suppose $\mathfrak{a}_i$ and $\mathfrak{a}'_i$ are consistent for every i, otherwise the consistency of $\mathfrak{b}, \mathfrak{b}'$ is trivial. Using Theorem 4.9, for each i choose some $\mathfrak{y}_i \in W_\sigma$ with $\mathfrak{y}_i \sqsupseteq \mathfrak{a}_i \sqcup \mathfrak{a}'_i$, so that $\mathfrak{y}_i$ represents $y_i \in T_{\sigma_i}$. Then $f(y_0) \cdots (y_{r-1}) = \mathfrak{f}(\mathfrak{y}_0) \cdots (\mathfrak{y}_{r-1}) = n$, and similarly $f(y_0) \cdots (y_{r-1}) = n'$, whence $n = n'$ and $\mathfrak{b}, \mathfrak{b}'$ are consistent.

(ii) By induction on types. The cases for $\overline{0}$ and for product types are easy, so assume the result for σ and τ, and suppose $\mathfrak{c} = \zeta_{\sigma\to\tau}(c)$ where $c = \langle a_0 \mapsto b_0, \ldots, a_{r-1} \mapsto b_{r-1}\rangle$. First, suppose $f \in U_\mathfrak{c}$; then there is some $\mathfrak{f} \in W_{\sigma\to\tau}$ representing f. To show that $f \in O_{\sigma\to\tau,c}$, suppose $x \in O_{\sigma,a_i}$ for some $i < r$; we wish to show that $f(x) \in O_{\tau,b_i}$. By the induction hypothesis we have $O_{\sigma,a_i} = U_{\mathfrak{a}_i}$ where $\mathfrak{a}_i = \zeta_\sigma(a_i)$, so there is some $\mathfrak{x} \in W_\sigma$ representing x with $\mathfrak{x} \sqsupseteq \mathfrak{a}$. But now $\mathfrak{f}(\mathfrak{x}) \sqsupseteq \mathfrak{c}(\mathfrak{a}_i) \sqsupseteq \mathfrak{b}_i = \zeta_\tau(b_i)$ and $\mathfrak{f}(\mathfrak{x})$ represents $f(x)$, so $f(x) \in U_{\mathfrak{b}_i}$. So by the induction hypothesis again, $f(x) \in O_{\tau,b_i}$.

Conversely, suppose $f \in O_{\sigma\to\tau,c}$, and take any $\mathfrak{f} \in W_{\sigma\to\tau}$ representing f. It will suffice to show that $\mathfrak{f}$ and $\mathfrak{c}$ are consistent, for then $\mathfrak{f} \sqcup \mathfrak{c} \in W_{\sigma\to\tau}$, so $\mathfrak{f} \sqcup \mathfrak{c}$ also represents f and witnesses that $f \in U_\mathfrak{c}$. In fact, it is enough to show that any basic compact $(\mathfrak{a} \Rightarrow \mathfrak{b}) \sqsubseteq \mathfrak{f}$ is consistent with each $\mathfrak{a}_i \Rightarrow \mathfrak{b}_i$, where $\mathfrak{a}_i = \zeta_\sigma(a_i)$, $\mathfrak{b}_i = \zeta_\tau(b_i)$. Suppose $\mathfrak{a}$ and $\mathfrak{a}_i$ are consistent, otherwise $\mathfrak{a} \Rightarrow \mathfrak{b}$ and $\mathfrak{a}_i \Rightarrow \mathfrak{b}_i$ are trivially consistent. Using Theorem 4.9, take some $\mathfrak{x} \in W_\sigma$ with $\mathfrak{x} \sqsupseteq \mathfrak{a} \sqcup \mathfrak{a}_i$, and suppose $\mathfrak{x}$ represents $x \in T_\sigma$. Then $\mathfrak{f}(\mathfrak{x})$ represents $f(x)$ and $\mathfrak{f}(\mathfrak{x}) \sqsupseteq \mathfrak{b}$, so $f(x) \in U_\mathfrak{b}$. But also $f \in O_{\sigma\to\tau,c}$ and $x \in U_{\mathfrak{a}_i} = O_{\sigma,a_i}$, so $f(x) \in O_{\tau,b_i} = U_{\mathfrak{b}_i}$ by the induction hypothesis. Thus $U_\mathfrak{b} \cap U_{\mathfrak{b}_i}$ is inhabited, and so $\mathfrak{b}, \mathfrak{b}_i$ are consistent by (i). Hence $\mathfrak{a} \Rightarrow \mathfrak{b}$ and $\mathfrak{a}_i \Rightarrow \mathfrak{b}_i$ are consistent as required. □

5. Continuous and effective N-TPCAs

The pattern that emerges from the foregoing results is that the type structure obtained from an extensional collapse construction appears to depend solely on whether the NR-TPCA or NR-TPCAs in question are "full continuous" or "effective". This leads us to look for precise notions of *full continuous* NR-TPCA and *effective* NR-TPCA that will allow us to obtain theorems of roughly the following kind:

— If A is a full continuous NR-TPCA, then $\mathsf{EC}(A) \cong \mathsf{C}$.
— If A is a full continuous NR-TPCA and A' an effective sub-NR-TPCA of A, then $\mathsf{EC}(A; A') \cong \mathsf{C}^{\mathit{eff}}$.
— If A is an effective NR-TPCA then $\mathsf{EC}(A) \cong \mathsf{HEO}$.

We now propose some such definitions, which seem to us to capture the intuitive notions we have in mind, and which moreover fit well into the general framework of *applicative morphisms* as presented in (Longley 1999b; Longley 2007?). This will allow

us, in Section 5.4, to give precise statements of our main results. A discussion of specific instances of these results will be postponed to Section 10.1.

5.1. *Realizations of N-TPCAs*

We will start with the following general notions:

Definition 5.1. Suppose A and B are N-TPCAs, with $\widehat{0}, \widehat{1}, \ldots$ a choice of numerals in A, and $\widetilde{0}, \widetilde{1}, \ldots$ a choice of numerals in B.

(i) A *realization* $\Vdash$ *of* A *over* B consists of a mapping $\sigma \mapsto [\sigma]$ from types to types, plus a family of relations $\Vdash_\sigma \subseteq B_{[\sigma]} \times A_\sigma$ (one for each σ) with the following properties:

— For all $a \in A_\sigma$ there exists $b \in B_{[\sigma]}$ such that $b \Vdash_\sigma a$.

— For each σ, τ, there is an element $\mathbf{a}_{\sigma\tau} \in B_{[\sigma\to\tau]\to[\sigma]\to[\tau]}$ such that for all $a \in A_{\sigma\to\tau}$, $a' \in A_\sigma$, $b \in B_{[\sigma\to\tau]}$ and $b' \in B_{[\sigma]}$,

$$b \Vdash_{\sigma\to\tau} a \ \wedge\ b' \Vdash_\sigma a' \ \wedge\ a \cdot a'{\downarrow} \implies \mathbf{a}_{\sigma\tau} \cdot b \cdot b' \Vdash_\tau a \cdot a'$$

(Informally, "$\mathbf{a}_{\sigma\tau}$ *tracks* $\cdot_{\sigma\tau}$".)

— There is an element $\mathbf{d} \in B_{[\overline{0}]\to\overline{0}}$ such that for all $n \in \mathbb{N}$ and $b \in B_{[\overline{0}]}$,

$$b \Vdash_0 \widehat{n} \implies \mathbf{d} \cdot b = \widetilde{n}.$$

We say A is *realizable over* B if there exists a realization of A over B.

(ii) A realization $\Vdash$ is called *discrete* if $b \Vdash_\sigma a \ \wedge\ b \Vdash_\sigma a' \implies a = a'$.

(iii) $\Vdash$ is *single-valued* if $b \Vdash_\sigma a \ \wedge\ b' \Vdash_\sigma a \implies b = b'$.

(iv) $\Vdash$ is *type-respecting* if $[\sigma] = \sigma$ for all σ, and each $\mathbf{a}_{\sigma\tau}$ may be taken to be $(\lambda^* f^{\sigma\to\tau} x^\sigma . fx)$.

(v) $\Vdash$ is *untyped* if B is an untyped PCA (construed as an N-TPCA). In this case we necessarily have $B_{[\sigma]} = B$ for all σ, and we may suppress mention of $[\sigma]$.

In fact, all the realizations we need to consider in this paper will be either type-respecting or untyped, but our more general definition allows us to treat these two situations uniformly. (An untyped realization need not be type-respecting, since the canonical choice for $\mathbf{a}_{\sigma\tau}$ might not be appropriate.)

We will sometimes use letters γ, δ, ϵ for realizations when we wish to consider them more abstractly as "morphisms of N-TPCAs". In this case we will write $[-]_\gamma$ for the mapping on types associated with γ, and will often use the infix notation $a \Vdash^\gamma_\sigma b$ in place $\gamma_\sigma(a, b)$. We write $\gamma : A \longrightarrow\!\!\triangleright B$ to mean "γ is a realization of A over B".

A realization $\Vdash$ of A over B can be informally thought of as a "copy" of A within the category $\mathbf{Asm}(B)$ (or $\mathbf{Mod}(B)$ if $\Vdash$ is discrete). We will make this precise below in the case where A is a total type structure. Alternatively, the above definition can be phrased concisely using the framework of (Longley 1999b): a realization of A over B is an applicative morphism $\gamma : A \mathrel{-\!\!\!\to} B$ which moreover respects the natural numbers up to realizable isomorphism.

The condition involving $\mathbf{d}$ ensures that the representation of numerals in A is in some sense "standard". To see why this condition is needed, let $A = K_1^T$ for T some non-trivial Turing degree, and $B = K_1$. Intuitively, we do not expect A to be realizable over

B. However, it turns out that there is an applicative morphism $K_1^T \rightarrowtail K_1$ (see (Longley 1995, Chapter 3)), since in some sense we can simulate non-effective computations in K_1 (informally, by means of thunks which are waiting for an oracle to be supplied). This is of no use in practice, though, because we have no effective way of "decoding" the result of the computation to obtain its value as a natural number, and this deficiency precisely corresponds to the absence of a suitable element **d**.

The following facts are easily checked. Detailed proofs of (i) and (iv) for the untyped setting are given in (Longley 1995, Chapter 2).

Proposition 5.2. (i) Given $\gamma : A \longrightarrow\!\!\triangleright B$ and $\delta : B \longrightarrow\!\!\triangleright C$, there is a realization $\delta \circ \gamma : A \longrightarrow\!\!\triangleright C$ defined by

$$[\sigma]_{\delta\circ\gamma} \;=\; [[\sigma]_\gamma]_\delta, \qquad c \Vdash^{\delta\circ\gamma}_\sigma a \;\text{ iff }\; \exists b \in B_{[\sigma]_\gamma}.\, c \Vdash^{\delta}_{[\sigma]_\gamma} b \;\wedge\; b \Vdash^{\gamma}_\sigma a$$

(ii) For any non-trivial N-TPCA A, there is a canonical discrete, type-respecting realization $\epsilon^A : \mathsf{EC}^{\mathsf{C}}(A) \longrightarrow\!\!\triangleright A$ given by

$$a \Vdash^{\epsilon^A}_\sigma x \;\text{ iff }\; a \sim_\sigma a \text{ and } \beta([a]) = x$$

Moreover, if A' is a sub-N-TPCA of A, ϵ^A restricts to a realization $\epsilon^{A;A'}$ of $\mathsf{EC}^{\mathsf{C}}(A; A')$ over A'.

(iii) For any NR-TPCA B, there is a canonical single-valued, type-respecting realization $\alpha^B : \mathsf{NRC} \longrightarrow\!\!\triangleright B$ defined by

$$b \Vdash^{\alpha^B} e \;\text{ iff }\; b \;=\; \ll e \gg$$

(iv) Any realization $\gamma : A \longrightarrow\!\!\triangleright B$ induces a functor $\gamma_* : \mathbf{Asm}(A) \to \mathbf{Asm}(B)$ defined by

$$|\gamma_*(X)| \;=\; X, \qquad b \Vdash_{\gamma_*(X)} x \iff \exists a \in A_{\rho_X}.\, b \Vdash^{\gamma}_{\rho_X} a \;\wedge\; a \Vdash_X x,$$

$$\gamma_*(f : X \to Y) \;=\; f$$

The functor γ_* preserves finite products and the natural number object (up to isomorphism). Moreover, if γ is discrete, γ_* restricts to a functor $\mathbf{Mod}(A) \to \mathbf{Mod}(B)$.

In connection with part (iv), it is worth observing why γ_* preserves the natural number object. Suppose $\widehat{0}, \widehat{1}, \ldots$ and $\widetilde{0}, \widetilde{1}, \ldots$ are choices of numerals in A and B respectively, and N_A, N_B are the corresponding natural number objects in $\mathbf{Asm}(A), \mathbf{Asm}(B)$. Then the canonical map $\gamma_*(N_A) \to N_B$ is realized by the element **d** of Definition 5.1, and its inverse by the element

$$\mathbf{rec}_0 \cdot \mathit{zero} \cdot (\mathbf{k}_{00} \cdot (\mathbf{a}_{00} \cdot \mathit{suc})) \;\in\; B_{\overline{0}\to\overline{0}}$$

where $\mathit{zero} \vdash^\gamma_0 \widehat{0}$ and $\mathit{suc} \vdash^\gamma_1 \mathbf{suc}$.

We now propose the following definition:

Definition 5.3. An *effective N-TPCA* is an N-TPCA A equipped with a realization $A \longrightarrow\!\!\triangleright K_1$.

In view of the foregoing proposition, the notion of an effective N-TPCA is quite robust: for example, if B is any PCA realizable over K_1 (such as K_2^{eff}), then an N-TPCA is

effective iff it is realizable over B. It is easy to check that all the "effective models" mentioned earlier — $\mathsf{P}^{\mathit{eff}}, \mathsf{L}^{\mathit{eff}}, \mathbb{T}^{\omega\,\mathit{eff}}, \mathcal{P}\omega^{\mathit{eff}}, K_2^{\mathit{eff}}$ and of course K_1 itself — are effective N-TPCAs.

In addition, if $\mathcal{L}$ is any effective extension of NRComb or PCF as in Section 3.4, any term model of the form $\mathcal{L}/\!\sim$ is an effective N-TPCA, since a Gödel-numbering of terms yields a realization for this N-TPCA in a standard way.

As a notational convention, we will reserve the symbol $\vdash$ (and decorated variants thereof) specifically for realizations over K_1, using the symbol $\Vdash$ in all other contexts. We will write $\vdash^{K_2\,\mathit{eff}}$ for the standard realization $K_2^{\mathit{eff}} \longrightarrow\!\!\triangleright K_1$ defined by

$$m \vdash^{K_2\,\mathit{eff}} g \;\text{ iff }\; g = \phi_m$$

A particular standard realization $\vdash^{\mathsf{P}\mathit{eff}}$ of $\mathsf{P}^{\mathit{eff}} \longrightarrow\!\!\triangleright K_1$ will be introduced in Section 5.2 below.

It is perhaps not so immediately clear what the definition of a continuous N-TPCA ought to be. We propose the following:

Definition 5.4. A *continuous N-TPCA* is an N-TPCA A equipped with a realization $A \longrightarrow\!\!\triangleright K_2$.

The intuition is that in a continuous N-TPCA, each object embodies only a countable amount of information (and hence can be coded by an element of K_2); moreover all the relevant operations must act continuously with respect to this information content (and hence be realizable within K_2). In fact, the continuous N-TPCAs are precisely those that can be represented within the framework of Type Two Effectivity (Weihrauch 2000); the connections between this framework and realizability over K_2 are explained in (Bauer 2002).

A little experimentation reveals that all the familiar examples of "continuous PCAs" (*e.g.* $\mathcal{P}\omega, \mathbb{T}^\omega, \mathcal{B}$) are realizable over K_2, as are the N-TPCAs P, L. Indeed, our notion of continuous N-TPCA seems to cover all the examples we have in mind. Moreover, this notion is robust, in that the same class of N-TPCAs would arise if we replaced K_2 in the definition by any other PCA A such that A and K_2 are realizable over each other (for instance, $A = \mathcal{P}\omega$).

As a slightly less obvious example, the NR-TPCA NRC^∞ of Section 3.4 is realizable over K_2. Likewise, it can be shown that the structure Q of Section 3.4 is a continuous NR-TPCA. We omit the proofs of these facts, which are not technically required for our purposes.

We say a continuous N-TPCA is *full continuous* if, informally, it contains all set-theoretic functions $\mathbb{N} \to \mathbb{N}$ in a "standard" way. More formally:

Definition 5.5. Suppose A is a continuous N-TPCA, with realization $\Vdash\colon A \longrightarrow\!\!\triangleright K_2$. We say A is *full continuous* if

— for every $f : \mathbb{N} \to \mathbb{N}$, there is some $a \in A_1$ that represents f;
— moreover, a realizer for some such element a may be computed from f within K_2. More precisely, there is an element $\mathbf{h} \in K_2$ such that for all $f \in \mathbb{N}^\mathbb{N}$, we have $\mathbf{h} \odot f \Vdash_1 a$ for some $a \in A_1$ representing f.

The second half of the condition says that the first half holds constructively "inside K_2"; this appears to be essential for the proofs of our theorems.[‖] It is easy to check that the examples of continuous N-TPCAs mentioned above, including NRC^∞, are all full continuous N-TPCAs according to the above definition.

We also need the notion of an effective sub-N-TPCA of a full continuous N-TPCA. Recall here that we have a sub-PCA K_2^{eff} of K_2, whose carrier set is the set of total computable functions.

Definition 5.6. Suppose A is a full continuous N-TPCA with realization $\Vdash\colon A \longrightarrow\!\!\triangleright K_2$, and $\mathbf{a}_{\sigma\tau}, \mathbf{d}, \mathbf{h}$ are as in Definitions 5.1 and 5.5. Suppose too that A' is a sub-N-TPCA of A. We say A' is an *effective sub-N-TPCA* of A (with respect to $\Vdash$) if

— for all $a \in A'_\sigma$, there exists $f \in K_2^{\mathit{eff}}$ with $f \Vdash_\sigma a$;
— for all σ, τ, the elements $\mathbf{a}_{\sigma\tau}, \mathbf{d}, \mathbf{h}$ may be taken to be in K_2^{eff}.

The notion of an *effective sub-NR-TPCA* is defined analogously.

Suppose A' is an effective sub-N-TPCA of a full continuous A with respect to $\Vdash\colon A \longrightarrow\!\!\triangleright K_2$. Then $\Vdash$ restricts to a realization $A' \longrightarrow\!\!\triangleright K_2^{\mathit{eff}}$, which we denote by $\Vdash_{\restriction A'}$. Since K_2^{eff} is itself an effective PCA, it follows that A' is an effective N-TPCA in its own right. In addition, the induced functor $\Vdash_*\colon \mathbf{Asm}(A) \to \mathbf{Asm}(K_2)$ restricts to a functor $\Vdash_*\colon \mathbf{Asm}(A; A') \to \mathbf{Asm}(K_2; K_2^{\mathit{eff}})$, which again preserves finite products and the natural number object.

5.2. *Some realizations of* P *and* $\mathsf{P}^{\mathit{eff}}$

We now introduce some particular realizations of the NR-TPCAs P and $\mathsf{P}^{\mathit{eff}}$ that will play an important role in later sections. Here we make heavy use of the concepts and notation introduced in Section 3.3.

Using this machinery, we may give a continuous realization of P as follows:

Definition 5.7. Define a realization $\Vdash^{\mathsf{P}}\colon \mathsf{P} \longrightarrow\!\!\triangleright K_2$ by

$$g \Vdash^{\mathsf{P}}_\sigma \mathfrak{u} \;\text{ iff }\; \forall i.\, g(i) \in \operatorname{dom} \zeta_\sigma \text{ and } \{\zeta_\sigma(g(i)) \mid i \in \mathbb{N}\} = C_{\mathfrak{u}}$$

That is, a realizer for $\mathfrak{u}$ is any function that enumerates precisely the set of compacts $\mathfrak{c} \sqsubseteq \mathfrak{u}$. We call $\Vdash^{\mathsf{P}}$ the *standard realization of* P.

To see that the application operations in P are indeed tracked by elements of K_2, first note that (by standard domain theory) for any $\mathfrak{f} \in \mathsf{P}_{\sigma\to\tau}, \mathfrak{u} \in \mathsf{P}_\sigma$, $C_{\mathfrak{f}(\mathfrak{u})}$ coincides with the set $\{\mathfrak{a}(\mathfrak{b}) \mid \mathfrak{a} \in C_{\mathfrak{f}}, \mathfrak{b} \in C_{\mathfrak{u}}\}$. Hence, for any σ, τ, using the function $\mathrm{apply}_{\sigma\tau}$ it is straightforward to construct an element $\mathbf{a}_{\sigma\tau} \in K_2^{\mathit{eff}}$ such that if $g \Vdash^{\mathsf{P}}_{\sigma\to\tau} \mathfrak{f}$ and $h \Vdash^{\mathsf{P}}_\sigma \mathfrak{u}$, then $\mathbf{a}_{\sigma\tau} \cdot g \cdot h \Vdash^{\mathsf{P}}_\tau \mathfrak{f}(\mathfrak{u})$. Moreover, it is easy to define an element $\mathbf{d} \in K_2^{\mathit{eff}}$ as required by Definition 5.1, so $\Vdash^{\mathsf{P}}$ is indeed a realization as claimed.

Furthermore, an element of P is effective iff it has at least one effective $\Vdash^{\mathsf{P}}$-realizer. Since the elements $\mathbf{a}_{\sigma\tau}$ and $\mathbf{d}$ above are effective, $\Vdash^{\mathsf{P}}$ restricts to a realization $\Vdash^{\mathsf{P}\mathit{eff}}$:

[‖] More abstractly, the fullness condition says that, in $\mathbf{Asm}(K_2)$, the evident morphism $\Vdash_*\,(\mathbf{EC}(A))_1 \to N^N$ has a right inverse, making N^N a retract of $\Vdash_*\,(\mathbf{EC}(A))_1$.

$\mathsf{P}^{\mathit{eff}} \longrightarrow\!\!\triangleright K_2^{\mathit{eff}}$. We may now define a standard effective realization of $\mathsf{P}^{\mathit{eff}}$ as promised earlier: let $\vdash^{\mathsf{P}\mathit{eff}} = \vdash^{K_2\,\mathit{eff}} \circ \Vdash^{\mathsf{P}\mathit{eff}}$.

5.3. *Realizations of type structures*

Since every total type structure is an N-TPCA, we immediately have the notion of a realization of a TTS over an N-TPCA. Examples include the realizations $\epsilon^A : \mathsf{EC}^{\mathsf{C}}(A) \longrightarrow\!\!\triangleright A$ given by Proposition 5.2(ii). It is easy to see that any realization of a TTS T over a non-trivial N-TPCA B must be discrete: the element $\mathbf{d}$ from Definition 5.1 shows that it is discrete at type $\overline{0}$, and the extensionality condition in the definition of TTS then implies discreteness at all types.

A realization γ of a total type structure T over an N-TPCA B corresponds exactly to an internal total type structure $\mathbf{T}^\gamma$ in $\mathbf{Asm}(B)$ (in the sense of Section 4) such that $|\mathbf{T}^\gamma| = T$ (where $|\mathbf{T}^\gamma|_\sigma = |\mathbf{T}^\gamma_\sigma|$, etc.). In the case $\gamma = \epsilon^B$, $\mathbf{T}^\gamma$ is precisely the structure we have called $\mathbf{EC}(B)$. The formulation in terms of internal TTSs makes particularly clear the notion of an *isomorphism* between realizations:

Definition 5.8. (i) Suppose $\mathcal{C}$ has finite products and a standard natural number object, and $\mathbf{T}, \mathbf{T}'$ are internal TTSs in $\mathcal{C}$. We say $\mathbf{T}, \mathbf{T}'$ are *isomorphic* if there is a (necessarily unique) family of isomorphisms $\mathbf{T}_\sigma \cong \mathbf{T}'_\sigma$ in $\mathcal{C}$ that commute with the given isomorphisms $\mathbf{T}_0 \cong N \cong \mathbf{T}'_0$ and the projection and application morphisms.

(ii) Suppose γ, γ' are realizations of TTSs T, T' in B respectively. We say γ, γ' are isomorphic ($\gamma \cong \gamma'$) if the corresponding internal TTSs $\mathbf{T}^\gamma, \mathbf{T}^{\gamma'}$ are isomorphic in $\mathbf{Asm}(B)$.

In the situation of (ii), clearly T, T' are isomorphic as ordinary TTSs. In the case where $T = T'$, the above definition may be rephrased concretely as follows: γ, γ' are isomorphic if for each σ there are elements $\mathbf{u}_\sigma \in B_{[\sigma]_\gamma \to [\sigma]_{\gamma'}}$ and $\mathbf{v}_\sigma \in B_{[\sigma]_{\gamma'} \to [\sigma]_\gamma}$ such that

$$b \Vdash^\gamma_\sigma x \implies \mathbf{u}_\sigma \cdot b \Vdash^{\gamma'}_\sigma x, \qquad b \Vdash^{\gamma'}_\sigma x \implies \mathbf{v}_\sigma \cdot b \Vdash^\gamma_\sigma x$$

Isomorphisms of this kind are also examples of *applicative isomorphisms* in the framework of (Longley 1999b).

Note that if $\mathbf{T}$ is an internal TTS in $\mathbf{Asm}(B)$ and we have a realization $\delta : B \longrightarrow\!\!\triangleright C$, then since by Proposition 5.2(iv) the induced functor $\delta_* : \mathbf{Asm}(B) \to \mathbf{Asm}(C)$ preserves finite products and the natural number object, we obtain an internal TTS $\delta_*(\mathbf{T})$ in $\mathbf{Asm}(C)$. If γ is a realization of T over B then clearly $\delta_*(\mathbf{T}^\gamma) = \mathbf{T}^{\delta \circ \gamma}$.

The internal TTS formulation also leads us naturally to "relative" versions of these concepts. Suppose B' is a sub-N-TPCA of B. A realization γ of T in B is called a *realization relative to* B' if the corresponding internal TTS $\mathbf{T}^\gamma$ in fact lies within $\mathbf{Asm}(B; B')$ (that is, the application morphisms are tracked by elements of B'). If γ, γ' are realizations of T, T' respectively in B relative to B', we say γ, γ' are *isomorphic relative to* B' if $\mathbf{T}^\gamma, \mathbf{T}^{\gamma'}$ are isomorphic in $\mathbf{Asm}(B; B')$ in the sense of Definition 5.8(i). Finally, if B' is an effective sub-N-TPCA of a full continuous B with respect to some $\delta : B \to K_2$, then any internal TTS $\mathbf{T}$ in $\mathbf{Asm}(B; B')$ gives rise to an internal TTS $\delta_*(\mathbf{T}) \in \mathbf{Asm}(K_2; K_2^{\mathit{eff}})$.

Our three type structures have the following natural realizations, which we may regard as the "standard" ones:

Definition 5.9. (i) The *standard realization* $\Vdash^{\mathsf{C}}$ *of* C is $\epsilon^{K_2} : \mathsf{EC}(K_2) \longrightarrow\!\!\triangleright K_2$. We write $\mathbf{C}$ for the corresponding internal TTS $\mathbf{EC}(K_2)$ in $\mathbf{Asm}(K_2)$.

(ii) The realization $\Vdash^{\mathsf{RC}}$ of $\mathsf{C}^{\mathit{eff}}$ is $\epsilon^{K_2;K_2^{\mathit{eff}}} : \mathsf{EC}(K_2;K_2^{\mathit{eff}}) \longrightarrow\!\!\triangleright K_2^{\mathit{eff}}$. We write $\mathbf{C}^{\mathit{eff}}$ for the corresponding internal TTS $\mathbf{EC}(K_2;K_2^{\mathit{eff}})$ in $\mathbf{Asm}(K_2;K_2^{\mathit{eff}})$.

(iii) The realization $\vdash^{\mathsf{HEO}}$ of HEO or $\mathsf{C}^{\mathit{heff}}$ is $\epsilon^{K_1} : \mathsf{EC}(K_1) \longrightarrow\!\!\triangleright K_1$. We write $\mathbf{HEO}$ for the corresponding internal TTS $\mathbf{EC}(K_1)$ in $\mathbf{Asm}(K_1)$.

The fact that these are indeed realizations of C, $\mathsf{C}^{\mathit{eff}}$, $\mathsf{C}^{\mathit{heff}}$ respectively depends on the equivalences between the relevant presentations of these type structures. This and more is given by the following theorem, which we shall require for our main proofs.

Theorem 5.10. (i) The realization $\Vdash^{\mathsf{P}} \circ \epsilon^{\mathsf{P}} : \mathsf{EC}(\mathsf{P}) \longrightarrow\!\!\triangleright K_2$ is isomorphic to $\Vdash^{\mathsf{C}}$.

(ii) The isomorphism in (i) holds relative to K_2^{eff}, and restricts to an isomorphism over K_2^{eff} between $\Vdash^{\mathsf{P}^{\mathit{eff}}} \circ \epsilon^{\mathsf{P};\mathsf{P}^{\mathit{eff}}}$ and $\Vdash^{\mathsf{RC}}$.

(iii) The realization $\vdash^{\mathsf{P}^{\mathit{eff}}} \circ \epsilon^{\mathsf{P}^{\mathit{eff}}} : \mathsf{EC}(\mathsf{P}^{\mathit{eff}}) \longrightarrow\!\!\triangleright K_1$ is isomorphic to $\vdash^{\mathsf{HEO}}$.

Part (iii), which corresponds to a strengthened version of the generalized KLS theorem, appears in (Berger 1993). Parts (i) and (ii) do not seem to have appeared explicitly in the literature, although a closely related fact, with L in place of P, can be read off from the results of (Bauer 2000; Bauer 2002). We include our own proof here, which is of some independent interest in that it seems to offer a relatively direct and perspicuous proof of the isomorphism $\mathsf{EC}(\mathsf{P}) \cong \mathsf{EC}(K_2)$.

Proof of Theorem 5.10. (i) We wish to prove that $\Vdash^{\mathsf{P}}_* (\mathbf{EC}(\mathsf{P})) \cong \mathbf{EC}(K_2)$ in $\mathbf{Asm}(K_2)$. We will construct the isomorphism for the pure types; the result for arbitrary types may be obtained similarly, or else by an appeal to Theorem 6.6 below.

For type $\bar{0}$, recall from Example 3.2 that each $n \in \mathbb{N}$ is represented in K_2 by the function $\widehat{n} : 0 \mapsto n,\ i+1 \mapsto 0$, and from Definition 5.7 that $n \in \mathbb{N} \subset \mathsf{P}_0$ is $\Vdash^{\mathsf{P}}$-realized by $h \in K_2$ iff $\mathrm{range}(h) = \{\breve{\bot}, \breve{n}\}$. It is easy to construct realizers $\alpha_0, \beta_0 \in K_2$ such that $\alpha_0 \odot h = \widehat{n}$ whenever $h \Vdash^{\mathsf{P}} n$, and $\beta_0 \odot \widehat{n} \Vdash^{\mathsf{P}} n$, showing that $\Vdash^{\mathsf{P}}_* (\mathbf{EC}(\mathsf{P}))_0 \cong \mathbf{EC}(K_2)_0$.

For the induction step, suppose $\imath_{k-1}$ is an isomorphism $\Vdash^{\mathsf{P}}_* (\mathbf{EC}(\mathsf{P}))_{k-1} \to \mathbf{EC}(K_2)_{k-1}$ realized by $\alpha_{k-1} \in K_2$, with inverse $\jmath_{k-1}$ realized by $\beta_{k-1} \in K_2$. To construct $\imath_k$, note that we have a morphism

$$\imath_0 \circ app_{k-1} \circ (id \times \jmath_{k-1}) \ : \ \Vdash^{\mathsf{P}}_* (\mathbf{EC}(\mathsf{P}))_k \times \mathbf{EC}(K_2)_{k-1} \longrightarrow \mathbf{EC}(K_2)_0$$

whose exponential transpose gives us a morphism $\imath_k : \Vdash^{\mathsf{P}}_* (\mathbf{EC}(\mathsf{P}))_k \longrightarrow \mathbf{EC}(K_2)_k$ which commutes with application.

To construct $\jmath_k$, consider an arbitrary morphism $\Phi : \mathbf{EC}(K_2)_{k-1} \to N$ in $\mathbf{Asm}(K_2)$ realized by some $h \in K_2$. We wish to show that Φ corresponds to a continuous function $\mathfrak{Phi} : \mathsf{P}_{k-1} \to \mathsf{P}_0$ such that $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}_k)$, and moreover that an enumeration ν of the compacts below $\mathfrak{Phi}$ may be computed from h within K_2. We will give an informal account of the construction of ν from h, from which it will be sufficiently clear that the construction could be performed within K_2 itself.

Given $h \Vdash^{\mathsf{C}}_k \Phi$, we may search for numbers n and finite sequences $p_0, \ldots, p_{r-1}$ such that $h(\langle 0, p_0, \ldots, p_{r-1}\rangle) = n+1$ for some n, but $h(\langle 0, p_0, \ldots, p_{t-1}\rangle) = 0$ for all $t < r$.

For any such $n, p_0, \ldots, p_{r-1}$, if $g \in K_2$ satisfies $g(i) = p_i$ for all $i < r$ and $h \odot g\!\downarrow$ then $(h \odot g)(0) = n$. Next, for each such $n, p_0, \ldots, p_{r-1}$ discovered, we may (in parallel) search for a sequence $q_0, \ldots, q_{s-1}$ such that:

— each q_j is a ζ_{k-1} code and the elements $\zeta_{k-1}(q_0), \ldots, \zeta_{k-1}(q_{s-1})$ are consistent;
— for each $i < r$ there exists $s_i \leq s$ such that $\alpha_{k-1}(\langle i, q_0, \ldots, q_{s_i-1}\rangle) = p_i + 1$, but $\alpha_{k-1}(\langle i, q_0, \ldots, q_{t-1}\rangle) = 0$ for all $t < s_i$.

For any such $q_0, \ldots, q_{s-1}$, if $\mu \in K_2$ satisfies $\mu(j) = q_j$ for all $j < s$ and $\alpha_{k-1} \odot \mu\!\downarrow$ then $(\alpha_{k-1} \odot \mu)(i) = p_i$ for all $i < r$.

For each such $n, p_0, \ldots, p_{r-1}, q_0, \ldots, q_{s-1}$ discovered, we output $\langle\langle q_0, \ldots, q_{s-1}\rangle \mapsto \check{n}\rangle$ as an element of the set enumerated by ν. We also (in parallel) output codes for all consistent finite joins of elements previously output, as well as codes for all compact elements below these. Finally, we may periodically output a code for $\bot_k$ in order to ensure that the output is indeed an infinite sequence of codes.

We claim that if $n, p_0, \ldots, p_{r-1}, q_0, \ldots, q_{s-1}$ are as above and $\mathfrak{b} = \bigsqcup_{j<s} \zeta_{k-1}(q_j)$, then for any $G \in U^{\mathsf{C}}_{\mathfrak{b}}$ we have $\Phi(\imath_{k-1}(G)) = n$. Indeed, given any such G, by definition of $U^{\mathsf{C}}_{\mathfrak{b}}$ there exists some $\mathfrak{G} \in \mathsf{Tot}(\mathsf{P}_{k-1})$ representing G with $\mathfrak{b} \sqsubseteq \mathfrak{G}$; thus there is some $\mu \Vdash^{\mathsf{P}}_{k-1} \mathfrak{G}$ with $\mu(j) = q_j$ for all $j < s$. But now μ realizes $G \in \Vdash^{\mathsf{P}}_{*} (\mathbf{EC}(P))_{k-1}$; hence $\alpha_{k-1} \odot \mu$ is defined and realizes $\imath_{k-1}(G) \in \mathbf{EC}(K_2)_{k-1}$. Hence $h \odot (\alpha_{k-1} \odot \mu)$ is defined and realizes $\Phi(\imath_{k-1}(G))$. But by the above observations, we have $(\alpha_{k-1} \odot \mu)(i) = p_i$ for $i < r$, whence $(h \odot (\alpha_{k-1} \odot \mu))(0) = n$, so $\Phi(\imath_{k-1}(G)) = n$.

It is now easy to see that the basic codes $\langle\langle q_0, \ldots, q_{s-1}\rangle \mapsto \check{n}\rangle$ arising from the above construction are all consistent. For if $\langle b \mapsto n\rangle$ and $\langle b' \mapsto n'\rangle$ are two such codes and $\zeta_{k-1}(b)$, $\zeta_{k-1}(b')$ are consistent, then by the density theorem we may take $\mathfrak{G} \sqsupseteq \zeta_{k-1}(b) \sqcup \zeta_{k-1}(b')$ representing some $G \in \mathsf{C}_{k-1}$, and then $n = \Phi(\imath_{k-1}(G)) = n'$. It now follows that *all* the compact elements enumerated by ν are consistent, so that $\bigsqcup_i \nu(i)$ defines an element $\mathfrak{Phi} \in \mathsf{P}_k$.

It remains to check that $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}_k)$ and that $\mathfrak{Phi}$ agrees with Φ. Take any $\mathfrak{G} \in \mathsf{Tot}(\mathsf{P}_{k-1})$ representing some $G \in \mathsf{C}_{k-1}$, and consider any $\mu \Vdash^{\mathsf{P}}_{k-1} \mathfrak{G}$. As before, we have $\alpha_{k-1} \odot \mu \Vdash^{\mathsf{C}}_{k-1} \imath_{k-1}(G)$ and $h \odot (\alpha_{k-1} \odot \mu) \Vdash^{\mathsf{C}}_{0} n = \Phi(\imath_{k-1}(G))$. So by continuity of application in K_2, there exist $p_0, \ldots, p_{r-1}$ and $q_0, \ldots, q_{s-1}$ satisfying the above conditions with $p_i = (\alpha_{k-1} \odot \mu)(i)$ for $i < r$ and $q_j = \mu(j)$ for $j < s$. Thus, the code $\langle\langle q_0, \ldots, q_{s-1}\rangle \mapsto \check{n}\rangle$ will turn up in ν, and since $\mu \Vdash^{\mathsf{P}}_{k-1} \mathfrak{G}$ it follows that $\mathfrak{Phi}(\mathfrak{G}) = n$ as desired.

For part (ii), it suffices to note that the constructions in the above proof are effective, so that the realizers α_k, β_k may all be chosen from K_2^{eff}.

Part (iii) is proved in (Berger 1993) in the form we require. The key ingredients of a proof of (iii) will also appear in the proof of Proposition 8.4 below. □

We conclude this section with a technical observation that will simplify our work later on: it is harmless to replace our realizations of P, $\mathsf{P}^{\mathit{eff}}$ with ones that involve only *basic* compact elements. Define relations $\Vdash^{\mathsf{P}}_{\mathrm{B}\sigma} \subseteq K_2 \times \mathsf{P}_\sigma$ by

$$g \Vdash^{\mathsf{P}}_{\mathrm{B}\sigma} \mathfrak{u} \;\text{ iff }\; \text{each } g(i) \text{ is a basic } \zeta_\sigma\text{-code and } \{\zeta_\sigma(g(i)) \mid i \in \mathbb{N}\} = B_{\mathfrak{u}}$$

In this situation, we will say that g *basically enumerates* $\mathfrak{u}$. (For example, in the context of Theorem 4.9, the function $\xi_{\sigma,c}$ basically enumerates $\mathfrak{x}_c$.)

Note that $\Vdash^{\mathsf{P}}_{\mathrm{B}}$ is not quite a realization of P over K_2, since there is nothing to realize the elements $\bot_\sigma \in \mathsf{P}_\sigma$; we will call $\Vdash^{\mathsf{P}}_{\mathrm{B}}$ a *near-realization* of P over K_2. However, we can see that $\Vdash^{\mathsf{P}}_{\mathrm{B}\sigma}$ is "isomorphic" to $\Vdash^{\mathsf{P}}_{\sigma}$ if we discount the bottom elements:

Proposition 5.11. For each σ, there are elements $\mathbf{u}_\sigma, \mathbf{v}_\sigma \in K_2^{\mathit{eff}}$ such that for all $g \in K_2$ and $\mathfrak{u} \in \mathsf{P}_\sigma$ with $\mathfrak{u} \neq \bot$,

$$g \Vdash^{\mathsf{P}}_{\mathrm{B}\sigma} \mathfrak{u} \implies \mathbf{u}_\sigma \cdot g \Vdash^{\mathsf{P}}_{\sigma} \mathfrak{u}, \qquad g \Vdash^{\mathsf{P}}_{\sigma} \mathfrak{u} \implies \mathbf{v}_\sigma \cdot g \Vdash^{\mathsf{P}}_{\mathrm{B}\sigma} \mathfrak{u}$$

Proof. From any enumeration g of $B_{\mathfrak{u}}$ we may obtain an enumeration $\overline{g}$ of $C_{\mathfrak{u}}$ by computing all existing finite joins of elements of $B_{\mathfrak{u}}$. Conversely, if g enumerates $C_{\mathfrak{u}}$ where $\mathfrak{u} \neq \bot$, let k be the least number such that $\zeta_\sigma(g(k)) \in B_{\mathfrak{u}}$, and define an enumeration $\widetilde{g}$ of $B_{\mathfrak{u}}$ by

$$\widetilde{g}(i) \;=\; \begin{cases} \mathrm{basic}_\sigma(g(i)) & \text{if } \zeta_\sigma(g(i)) \text{ is basic,} \\ \mathrm{basic}_\sigma(g(k)) & \text{otherwise} \end{cases}$$

Clearly the mappings $g \mapsto \overline{g}$ and $g \mapsto \widetilde{g}$ are both representable within K_2^{eff}. □

It follows easily from this that the composition $\Vdash^{\mathsf{P}}_{\mathrm{B}} \circ \epsilon^{\mathsf{P}}$ (defined exactly as in Proposition 5.2(i)) is a genuine realization of $\mathsf{EC}(\mathsf{P})$ over K_2 which is isomorphic to $\Vdash^{\mathsf{P}} \circ \epsilon^{\mathsf{P}}$. Moreover, $\Vdash^{\mathsf{P}}_{\mathrm{B}}$ restricts to a near-realization $\Vdash^{\mathsf{P}^{\mathit{eff}}}_{\mathrm{B}}$ of $\mathsf{P}^{\mathit{eff}}$ over K_2^{eff}, and $\Vdash^{\mathsf{P}^{\mathit{eff}}}_{\mathrm{B}} \circ \epsilon^{\mathsf{P};\mathsf{P}^{\mathit{eff}}}$ is then a genuine realization of $\mathsf{EC}(\mathsf{P};\mathsf{P}^{\mathit{eff}})$ over K_2^{eff} which is isomorphic to $\Vdash^{\mathsf{P}^{\mathit{eff}}} \circ \epsilon^{\mathsf{P};\mathsf{P}^{\mathit{eff}}}$. Finally, we may define a near-realization $\vdash^{\mathsf{P}^{\mathit{eff}}}_{\mathrm{B}} = \vdash^{K_2 \mathit{eff}} \circ \Vdash^{\mathsf{P}^{\mathit{eff}}}_{\mathrm{B}}$ of $\mathsf{P}^{\mathit{eff}}$ over K_1, and then $\vdash^{\mathsf{P}^{\mathit{eff}}}_{\mathrm{B}} \circ \epsilon^{\mathsf{P}^{\mathit{eff}}}$ is a genuine realization of $\mathsf{EC}(\mathsf{P}^{\mathit{eff}})$ over K_1 which is isomorphic to $\vdash^{\mathsf{P}^{\mathit{eff}}} \circ \epsilon^{\mathsf{P}^{\mathit{eff}}}$. Thus, Theorem 5.10 also holds with $\Vdash^{\mathsf{P}}, \Vdash^{\mathsf{P}^{\mathit{eff}}}, \vdash^{\mathsf{P}^{\mathit{eff}}}$ replaced by $\Vdash^{\mathsf{P}}_{\mathrm{B}}, \Vdash^{\mathsf{P}^{\mathit{eff}}}_{\mathrm{B}}, \vdash^{\mathsf{P}^{\mathit{eff}}}_{\mathrm{B}}$ respectively.

5.4. *The main theorems*

We are now at last able to give precise statements of our main results. For technical reasons, all three of our theorems require certain hygiene conditions (which in the case of C and $\mathsf{C}^{\mathit{eff}}$ were missing in (Longley 2005b)). These conditions make the statements of our theorems less elegant than we would like, and exclude some interesting examples, but we cannot at present see how to prove our theorems under significantly weaker hypotheses.

It is convenient to state the theorem for the effective case first:

Theorem 5.12 (Ubiquity of HEO). Let A be an effective NR-TPCA, with γ a realization of A over K_1 satisfying the following conditions:

(A) If $m \vdash^{\gamma}_{\sigma\to\overline{0}} a$, $n \vdash^{\gamma}_{\sigma} b$ and $\mathbf{a}_{\sigma 0} \bullet m \bullet n \vdash^{\gamma}_{0} c$ (where $\mathbf{a}_{\sigma 0}$ tracks $\cdot_{\sigma 0}$ as in Definition 5.1), then $a \cdot b = c$.

(B) The set $\{m \mid m \vdash^{\gamma}_{0} \widehat{0}\}$ is c.e.

Then $\gamma_*(\mathbf{EC}(A)) \cong \mathbf{HEO}$ in $\mathbf{Asm}(K_1)$. (That is, $\mathsf{EC}(A) \cong \mathsf{HEO}$, and the realizations $\gamma \circ \epsilon^A$ and $\vdash^{\mathsf{HEO}}$ over K_1 are isomorphic.)

Condition (A) here is very mild, and in all naturally arising models that we know, it can be arranged to hold simply by choosing the realizers $\mathbf{a}_{\sigma 0}$ suitably. Condition (B) holds

in most of the known models, including the term models for all reasonable deterministic programming languages extending NRComb, and the effective substructures of all reasonable domain-theoretic models in which the type of natural numbers is modelled by the usual domain $N_\perp$. Unfortunately, it rules out a few important models such as K_2^{eff} and $\mathcal{P}\omega^{\mathit{eff}}$, although in these cases the conclusion is already known to hold anyway. It also excludes the term models for certain non-deterministic programming languages, and the effective Nakajima tree model of the untyped lambda calculus (see (Barendregt 1984)). We have so far been unable to prove a version of the above theorem that embraces these additional models.

Clearly, conditions (A) and (B) taken together imply the following somewhat more artificial condition:

(AB) For any type σ, element $a \in A_{\sigma\to\overline{0}}$ and realizer $m \vdash^{\gamma}_{\sigma\to\overline{0}} a$, there is a c.e. set $R \subseteq \mathbb{N}$ such that whenever $n \vdash^{\gamma}_{\sigma} b$, we have $a \cdot b = \widehat{0}$ if and only if $n \in R$.

In order to state the corresponding theorem for C, we have to use the analogue of condition (AB) rather than of (A) and (B) separately. (This is because of the annoying fact that the definedness of application in K_2 is not semidecidable — see Example 3.2). We also require a further condition (C) which we shall explain below.

In order to formulate (AB) in the continuous setting, we replace the notion of computable enumerability by the topological notion of *openness*. A set $R \subseteq \mathbb{N}^{\mathbb{N}}$ is open if for any $f \in R$ there exists i such that for any $f' \in \mathbb{N}^{\mathbb{N}}$, if $\widetilde{f'}(i) = \widetilde{f}(i)$ then $f' \in R$.

Theorem 5.13 (Ubiquity of C). Let A be any full continuous NR-TPCA, with γ a realization of A over K_2 satisfying the following conditions:

(AB) For any type σ and any $f \Vdash^{\gamma}_{\sigma\to\overline{0}} a$, there is an open subset $R \subseteq \mathbb{N}^{\mathbb{N}}$ such that whenever $g \Vdash^{\gamma}_{\sigma} b$, we have $a \cdot b = \widehat{0}$ if and only if $g \in R$.

(C) There is an element $\mathbf{norm} \in A_{\overline{1}\to\overline{1}}$ (called a *normalizer* for type 1 elements) such that whenever $\dot{f}, \dot{f}' \in A_1$ both represent some $f : \mathbb{N} \to \mathbb{N}$, the element $\mathbf{norm} \cdot \dot{f}$ also represents f and $\mathbf{norm} \cdot \dot{f} = \mathbf{norm} \cdot \dot{f}'$.

Then $\gamma_*(\mathbf{EC}(A)) \cong \mathbf{C}$ in $\mathbf{Asm}(K_2)$. (That is $\mathsf{EC}(A) \cong \mathsf{C}$, and the realizations $\gamma \circ \epsilon^A$ and $\Vdash^{\mathsf{C}}$ over K_2 are isomorphic.)

The same conditions also suffice for the theorem for $\mathsf{C}^{\mathit{eff}}$. We recall the notion of an effective sub-NR-TPCA from Definition 5.6.

Theorem 5.14 (Ubiquity of $\mathsf{C}^{\mathit{eff}}$). Let A be any full continuous TPCA, with γ a realization of A over K_2, such that conditions (AB) and (C) of Theorem 5.13 hold, and let A' be an effective sub-NR-TPCA of A with respect to γ, which also contains the element **norm** from condition (C). Then $\gamma_*(\mathbf{EC}(A; A')) \cong \mathbf{C}^{\mathit{eff}}$ in $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$.

Once again, the condition (AB) holds in most of the settings of interest, but excludes the models K_2, $\mathcal{P}\omega$ and the full Nakajima tree model.

Condition (C) says that the representation of type 1 functions in A is not too intensional: although many elements of A_1 may represent the same type 1 function f, we may

pass to a canonical representative of f within A itself.[††] This property typically holds in any moderately denotational setting, but fails in some "syntactic" models such as NRC^∞. Although one might prefer to dispense with this condition, it is perhaps not too surprising that it should play a role when the realizing objects are themselves extensional type 1 functions (elements of K_2). It seems to us that condition (C) would be much less acceptable in the effective setting, where it would rule out the term models for many programming languages of interest. (For further discussion of the scope and limitations of our theorems, see Section 10.1.)

In the presence of condition (C), the element $\mathbf{h}$ witnessing the fullness of A (see Definition 5.5) may be chosen so that for any $f \in K_2$, $\mathbf{h} \odot f$ realizes a normalized element of A_1. Indeed, if $\mathbf{h} \in K_2$ is any realizer satisfying the condition of Definition 5.5, and $norm \in K_2$ is any realizer for **norm**, then we may take $\mathbf{h}' \in K_2$ such that $\mathbf{h}' \odot f \succeq \mathbf{a}_{11} \odot norm \odot (\mathbf{h} \odot f)$ for all $f \in K_2$; then $\mathbf{h}' \odot f$ realizes an element $a \in A_1$ such that a represents f and $\mathbf{norm} \cdot a = a$. We will say a realizer $\mathbf{h}$ is *normal* if it satisfies this additional property. We also write $f^\sharp$ for the canonical representative of f in A_1, so that $\mathbf{norm} \cdot f^\sharp = f^\sharp$.

6. Preliminaries to the main proofs

The proofs of our main theorems are rather long and technical, and make essential use of a technique developed in (Normann 2000). We begin with an informal overview of our method of proof, in order to highlight certain conceptual aspects which do not stand out so clearly from the proofs themselves.

6.1. *Conceptual overview*

We start by recalling Normann's theorems from (Normann 2000), glossing over a few subtleties for the sake of the broad picture. Normann's results were given in both a "continuous" and an "effective" version, and were presented in terms of the language PCF and its interpretation in the type structures P and $\mathsf{P}^{\mathit{eff}}$. As in Section 3.4, we will write PCF^∞ for the language obtained by extending PCF with a constant c_f for every $f : \mathbb{N} \to \mathbb{N}$. The relevant parts of Normann's theorems may then be stated in our terminology as follows:

Theorem 6.1 (Normann). (i) For every $\Phi \in \mathsf{C}_\sigma$, there is a PCF^∞-definable element $\dot{\Phi} \in \mathsf{Tot}(\mathsf{P}_\sigma)$ such that $[\dot{\Phi}] = \Phi$.

(ii) For every $\Phi \in \mathsf{HEO}_\sigma$, there is a PCF-definable element $\dot{\Phi} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_\sigma)$ such that $[\dot{\Phi}] = \Phi$.

To prove these theorems, it suffices to restrict attention to the pure types $\overline{k}$ (see Section 6.2). At the heart of Normann's proofs is the construction of a PCF program which, given a basic enumeration ν for some $\ddot{\Phi} \in \mathsf{Tot}(\mathsf{P}_k)$ representing Φ, "simulates" this functional so as to yield an element $\dot{\Phi}$ also representing Φ. Very crudely, the idea is

[††] More abstractly, condition (C) says that, in $\mathbf{Asm}(A)$, the object $\mathbf{EC}(A)_1$ is projective.

as follows: given some $\dot{G} \in \mathsf{Tot}(\mathsf{P}_{k-1})$ corresponding to $G \in \mathsf{C}_{k-1}$, to compute $\Phi(G)$ we search through the codes enumerated by ν, testing each code $c \mapsto \breve{q}$ in turn to see whether $G \in O_{k-1,c}$, and when such a code is found we simply return the number q. The evident problem is how to do this in such a way that each individual test is guaranteed to terminate, and this problem is overcome in Normann's proofs by an ingenious use of recursion. (An informal explanation of Normann's method at a more detailed level will be given in Section 7.5.)

At first sight, Normann's theorems seem quite specific to the models P and $\mathsf{P}^{\mathit{eff}}$. However, we can "detach" the theorem from these models to some extent by concentrating on the PCF$^\infty$ or PCF terms that define the elements $\dot{\Phi}$. Let PCF and PCF^∞ denote the term models for PCF and PCF$^\infty$ respectively as in Section 3.4. Then (concentrating on the continuous case) the familiar interpretation map $[\![-]\!]$ gives us a single-valued, type-respecting realization $\alpha : \mathsf{PCF}^\infty \longrightarrow\!\triangleright \mathsf{P}$, and Normann's proof associates to each $\Phi \in \mathsf{C}$ a corresponding set of PCF$^\infty$ terms, giving rise to a realization $\eta : \mathsf{C} \longrightarrow\!\triangleright \mathsf{PCF}^\infty$. The idea is that PCF^∞ contains a "copy" of C which can in principle be defined purely syntactically without reference to P. We may therefore view the situation in terms of a sequence of realizations

$$\mathsf{C} \xrightarrow{\ \eta\ } \mathsf{PCF}^\infty \xrightarrow{\ \alpha\ } \mathsf{P} \xrightarrow{\ \Vdash^{\mathsf{P}}_{\mathrm{B}}\ } K_2$$

where $\Vdash^{\mathsf{P}}_{\mathrm{B}}$ is the realization of P via basic enumerations (see Section 5.3).

Expressed in these terms, Normann's theorem tells us that if $\dot{\Phi} \Vdash^{\alpha\circ\eta} \Phi$ then $\dot{\Phi} \Vdash^{\epsilon^{\mathsf{P}}} \Phi$ (we may naturally express this by writing $\alpha \circ \eta \subseteq \epsilon^{\mathsf{P}}$). Clearly the converse does not hold, since not every element of $\mathsf{Tot}(\mathsf{P})$ is PCF$^\infty$-definable, and in fact the realizations $\alpha \circ \eta$ and ϵ^{P} are not even isomorphic. However, these realizations *are* equivalent at the level of K_2 realizers — in fact, the three realizations $\Vdash^{\mathsf{P}}_{\mathrm{B}} \circ \alpha \circ \eta$, $\Vdash^{\mathsf{P}}_{\mathrm{B}} \circ \epsilon^{\mathsf{P}}$ and $\Vdash^{\mathsf{C}}\colon \mathsf{C} \longrightarrow\!\triangleright K_2$ are all isomorphic. The point is that a $\Vdash^{\mathsf{P}}_{\mathrm{B}}$-realizer for a general $\ddot{\Phi} \in \mathsf{Tot}(\mathsf{P})$ is just a basic enumeration of $\ddot{\Phi}$, and Normann's program gives us an effective way to pass from such an enumeration to a suitable PCF$^\infty$-definable element $\dot{\Phi}$.

Actually, it is only a short step from here to show that the extensional collapse of PCF^∞ itself is C (the argument essentially appears in (Plotkin 1997)). This fact is helpful for motivating what we are going to do, although technically our proof does not proceed via quite this route.

The situation for the effective case is exactly analogous, and the above discussion applies *mutatis mutandis* to the sequence of realizations

$$\mathsf{HEO} \xrightarrow{\ \eta\ } \mathsf{PCF} \xrightarrow{\ \alpha\ } \mathsf{P}^{\mathit{eff}} \xrightarrow{\ \vdash^{\mathsf{P}\mathit{eff}}_{\mathrm{B}}\ } K_1$$

Having seen how Normann's results fit into our framework of N-TPCAs and realizations, we now explain the two main respects in which our setting in the present paper is different from Normann's.

The first step is rather a minor one: we replace PCF by NRComb. As mentioned in Section 3.4, one may think of NRComb as a slightly weaker language than PCF, though the gap between the two languages should be thought of as relatively small. It turns out (as we shall see) that the Normann programs can equally well be expressed in NRComb.

In the effective case, we may simply replace the term model PCF in the above discussion by NRC (see Section 3.4). In the continuous case, we would like to be able to similarly replace PCF^∞ by NRC^∞. However, so far we have only able to carry out our proofs for the continuous case using a slightly richer language than NRComb^∞ (this corresponds to the need for condition (C) in Theorem 5.13). Let NRComb^+ be the language obtained from NRComb by adding a new constant $\mathtt{norm} : \overline{1} \to \overline{1}$, and let $\mathrm{NRComb}^{\infty+}$ be the analogous extension of NRComb^∞. We may extend the proof system for NRComb given in Section 3.4 by adding the infinitary rule:

— If for each $n \in \mathbb{N}$ we have $\vdash e_0\,\lceil n \rceil \downarrow$ and $\vdash e_0\,\lceil n \rceil \simeq e_1\,\lceil n \rceil$, then $\vdash \mathtt{norm}\, e_0 \downarrow$ and $\vdash \mathtt{norm}\, e_0 \simeq \mathtt{norm}\, e_1$.

We may thence obtain a term model $\mathsf{NRC}^{\infty+}$ as before.

The second step is much more radical: we replace P and $\mathsf{P}^{\mathit{eff}}$ in the above by an arbitrary continuous or effective NR-TPCA A, subject to the conditions of the relevant theorem. For instance, suppose A is any effective NR-TPCA. Then A comes equipped with a realization $\alpha : \mathsf{NRC} \longrightarrow\!\!\triangleright A$ simply by virtue of being an NR-TPCA (see Proposition 5.2(iii)), and a realization $\gamma : A \longrightarrow\!\!\triangleright K_1$ by virtue of being effective. Indeed, in terms of computational power, it is natural to think of NRC as the "weakest" effective NR-TPCA and K_1 as the "strongest", with A sandwiched between them.

For the continuous case, we must suppose A is a full continuous NR-TPCA satisfying condition (C) of Theorem 5.13; this means that we can naturally interpret $\mathrm{NRComb}^{\infty+}$ in A and so obtain a realization $\mathsf{NRC}^{\infty+} \longrightarrow\!\!\triangleright A$. The realization $A \longrightarrow\!\!\triangleright K_2$ comes from the hypothesis that A is continuous. Once again, we may think of $\mathsf{NRC}^{\infty+}$ and K_2 as the weakest and strongest continuous NR-TPCAs that we are considering.

Moreover, by earlier remarks, $\mathsf{NRC}^{\infty+}$ [resp. NRC] contains a copy of C [resp. HEO] arising from the Normann programs and embodied by the realization η. In fact, we have $\mathsf{EC}(\mathsf{NRC}^{\infty+}) \cong \mathsf{C}$ and $\eta \subseteq \epsilon^{\mathsf{NRC}^{\infty+}}$ [resp. $\mathsf{EC}(\mathsf{NRC}) \cong \mathsf{HEO}$ and $\eta \subseteq \epsilon^{\mathsf{NRC}}$]. To summarize our data, in the continuous case we have realizations

$$\mathsf{C} \xrightarrow{\ \eta\ \triangleright} \mathsf{NRC}^{\infty+} \xrightarrow{\ \alpha\ \triangleright} A \xrightarrow{\ \gamma\ \triangleright} K_2$$

while in the effective case we have

$$\mathsf{HEO} \xrightarrow{\ \eta\ \triangleright} \mathsf{NRC} \xrightarrow{\ \alpha\ \triangleright} A \xrightarrow{\ \gamma\ \triangleright} K_1$$

Thus (in the continuous case for example) the realization $\alpha \circ \eta$ shows that a copy of C exists within $\mathbf{Asm}(A)$. Indeed, A is sandwiched between two NR-TPCAs whose extensional collapse is C, and this already makes it seem plausible that $\mathsf{EC}(A) = \mathsf{C}$. Somewhat less crudely, one might hope to show by an induction up the types that $\gamma_*(\mathbf{EC}(A))_k \cong \mathbf{C}_k$ in $\mathbf{Asm}(K_2)$ for each k; at each level, the realization $\alpha \circ \eta$ would tell us that the set $\mathsf{EC}(A)_k$ contains *at least* the continuous functionals of type k, whilst γ would tell us that it contains *at most* these functionals. However, as mentioned in Section 4, there are examples of similar situations in which the extensional collapse construction behaves pathologically, and this warns us that we cannot hope for a simple proof of our theorem in terms of general abstract considerations.

In fact, an induction of the above kind does go through, and can be used to show that the internal TTSs $\mathbf{EC}(K_2)$, $\gamma_*(\mathbf{EC}(A))$ and $(\gamma\circ\alpha\circ\eta)_*(\mathsf{C})$ are all isomorphic in $\mathbf{Asm}(K_2)$, and moreover that $\alpha \circ \eta \subseteq \epsilon^A$. (It is *not* true in general that $\mathbf{EC}(A) \cong (\alpha \circ \eta)_*(\mathsf{C})$ in $\mathbf{Asm}(A)$.) The difficult part is to show the inclusion $\alpha\circ\eta \subseteq \epsilon^A$ at each type level — that is, to show that if $\dot{\Phi} \in A_k$ is the denotation in A of a Normann program for Φ, then $\dot{\Phi}$ acts as Φ on the whole of $\mathsf{Tot}(A_{k-1})$, not just the part of $\mathsf{Tot}(A_{k-1})$ consisting of the denotations of Normann programs. In other words, we may readily show that our Normann programs behave as expected when applied to other Normann programs, but why on earth should $\dot{\Phi}$ behave sensibly when applied to a "non-program" $\dot{G} \in \mathsf{Tot}(A_{k-1})$, about which we know nothing at all except that it acts extensionally on elements of $\mathsf{Tot}(A_{k-2})$?

Actually, the behaviour of $\dot{\Phi}$ on $\dot{G}$ will be quite predictable if the Normann program only "uses" $\dot{G}$ in certain restricted ways — in particular, if it only ever applies $\dot{G}$ to total elements $\dot{h} \in \mathsf{Tot}(A_{k-2})$ representing known elements of C_{k-2}. Indeed, in the case $k = 2$, this is just what happens and so it is easy to show that the Normann programs behave correctly (see Section 7.1). In general, however, the way the Normann programs work is more subtle. At each stage in our search through the enumeration ν, the program will "test" $\dot{G}$ by applying it to some $\dot{h} \in A_{k-2}$ which (recursively) depends on the result of searching further down the enumeration. Will this element $\dot{h}$ be total? The form of the Normann programs ensures that this will indeed be the case provided that some test further on in ν succeeds, which would seem to depend on the fact that the elements $\dot{h}'$ used in that test are total, and so on. The problem, then, is to find some way of breaking this apparent infinite regress. Once we have shown that at least *one* of the tests arising from ν returns true, we can use reversed induction to show that all previous stages of the computation are well-behaved, in that the elements $\dot{h}$ involved are all total so that $\dot{G} \cdot \dot{h}$ yields a meaningful value, and hence that the final result of the computation is indeed $\Phi(G)$.

In Normann's original proof, the existence of such an "end-stop" in ν was proved by appealing to the continuity of $\dot{G}$ in P: at some stage in our search we would encounter a test element $\dot{h}$ which, if not a total element, was a sufficiently good approximation to one that the test involving $\dot{G}\cdot\dot{h}$ would succeed. In our more general setting, such an approach is not directly applicable, since A need not have any kind of CPO structure or any overt notion of "partial approximation" to a total element. The key to our approach lies with the realization $\gamma : A \longrightarrow\!\!\!\!\triangleright K_2$: we can show that simply by virtue of being a continuous NR-TPCA, A inherits enough "continuity" from K_2 that an argument akin to Normann's can be used to show the existence of a suitable end-stop. In fact, a substantial portion of our proof (Sections 7.2–7.4) will be devoted to making this idea work; as we shall see, the task entails some significant elaborations of the Normann programs themselves. Once the existence of an end-stop has been established, our argument runs closely parallel to that of (Normann 2000) (Sections 7.5–7.6).

In the effective case, the strategy is similar, and our task is to show that an effective NR-TPCA A inherits some kind of continuity from K_1. Here we must understand "continuity in K_1" in the sense associated with the Myhill-Shepherdson and Kreisel-Lacombe-Shoenfield theorems, and indeed, ideas from the proofs of these theorems will play a vital role in our arguments. In this case, even more work is required to show

the existence of an end-stop (Sections 8.1–8.3); once this is done, the argument is again broadly similar to Normann's (Sections 8.4–8.5).

Whilst the term models NRC and $\mathsf{NRC}^{\infty+}$ play a key conceptual role, we will not henceforth refer to them explicitly in the proofs themselves, since to do so would impose additional notational complications which do not seem to justify themselves. Nevertheless, the reader may find it helpful to make the mental connection with the abstract picture outlined above.

6.2. *Reduction to pure types*

As with many other results in higher type computability, it will suffice to prove our main theorems for the pure types, since all other types arise as retracts of these. It is convenient to dispose of this issue in advance of tackling the main proofs, since it will significantly lighten our notational burden if we only have to work with pure types.

The encoding of arbitrary types in pure types has been worked out many times over in the literature (see for example (Troelstra 1973, §1.8)), but unfortunately, the results still do not seem to appear in quite the form we need. Indeed, it seems to us that the matter is not quite as simple as is sometimes supposed: one not only has to show that every type is a retract of a pure type, but also that the corresponding idempotents can be identified by looking at the pure types alone. Even so, there are no major surprises in this section, and readers who are already comfortable with these ideas may safely skip it.

Let T be any total type structure as defined in Section 4. We will show that the whole of T may be reconstructed from the *pure part* of T — that is, the family of sets T_k together with the application operations $T_{k+1} \times T_k \to \mathbb{N}$. Since every TTS is an N-TPCA, it is convenient to use the language NComb as a notation for uniformly specifying elements of T.

Recall from Section 2.1 that *definition environments* for NComb may be built up by cumulative sequences of defining equations. By a *defined constant* we shall mean simply a derived constant together with a definition environment of NComb in which it appears. Clearly, any defined constant has a canonical interpretation in any TTS. We shall sometimes refer to defined constants somewhat informally as *programs*.

One particular definition environment will be of special significance. Let Δ_0 be some environment that defines just three constants $\texttt{make-pair} : \overline{0}^{(2)} \to \overline{0}$, $\texttt{proj0} : \overline{0} \to \overline{0}$, $\texttt{proj1} : \overline{0} \to \overline{0}$ as in Section 2.2.

Let us say a type ρ is *regular* if it is of the form $\sigma_0 \to \cdots \to \sigma_{r-1} \to \overline{0}$ where all the σ_i are pure types. A defining equation

$$\mathtt{f}\,\mathtt{x}_0 \ldots \mathtt{x}_{r-1} \;\equiv\; e$$

is called regular if the variables $\mathtt{x}_i$ and e itself are all of pure type, and moreover all subterms of e are of regular type. We say a definition environment Δ is regular if it can be built up from Δ_0 entirely by means of regular equations, and a defined constant is regular if its associated definition environment is regular. Clearly, a regular defined constant must be of regular type.

Proposition 6.2. (i) For each k there exist regular defined constants

$$\mathtt{up}_k : \overline{k} \to \overline{k+1}, \qquad \mathtt{down}_k : \overline{k+1} \to \overline{k}$$

such that for any TTS T and for all $y \in T_k$ we have $down_k \cdot (up_k \cdot y) = y$.

(ii) For each k there are regular defined constants

$$\mathtt{Pair}_k : \overline{k} \to \overline{k} \to \overline{k}, \qquad \mathtt{Fst}_k : \overline{k} \to \overline{k}, \qquad \mathtt{Snd}_k : \overline{k} \to \overline{k}$$

such that for any TTS T and for all $z, z' \in T_k$ we have $Fst_k \cdot (Pair_k \cdot z \cdot z') = z$ and $Snd_k \cdot (Pair_k \cdot z \cdot z') = z'$.

Proof. (i) By induction on k, define

$$\begin{aligned} \mathtt{up}_0\ \mathtt{y}^0\ \mathtt{x}^0 &\equiv \mathtt{y} \\ \mathtt{down}_0\ \mathtt{z}^1 &\equiv \mathtt{z\ 0} \\ \mathtt{up}_{k+1}\ \mathtt{y}^{k+1}\ \mathtt{x}^{k+1} &\equiv \mathtt{y\ (down}_k\ \mathtt{x)} \\ \mathtt{down}_{k+1}\ \mathtt{z}^{k+2}\ \mathtt{w}^k &\equiv \mathtt{z\ (up}_k\ \mathtt{w)} \end{aligned}$$

An easy induction shows that $\mathtt{up}_k, \mathtt{down}_k$ have the required property.

(ii) For the case $k = 0$, let $\mathtt{Pair}_0 \equiv \mathtt{make\text{-}pair}$, $\mathtt{Fst}_0 \equiv \mathtt{proj}_0$, $\mathtt{Snd}_0 \equiv \mathtt{proj}_1$. For $k > 0$, define

$$\begin{aligned} \mathtt{Pair}_k\ \mathtt{z}^k\ \mathtt{z'}^k\ \mathtt{x}^{k-1} &\equiv \mathtt{Pair}_0\ (\mathtt{z\ x})\ (\mathtt{z'\ x}) \\ \mathtt{Fst}_k\ \mathtt{z}^k\ \mathtt{x}^{k-1} &\equiv \mathtt{Fst}_0\ (\mathtt{z\ x}) \\ \mathtt{Snd}_k\ \mathtt{z}^k\ \mathtt{x}^{k-1} &\equiv \mathtt{Snd}_0\ (\mathtt{z\ x}) \end{aligned}$$

Again, it is easy to check that these constants have the required property. □

For $k > 0$, let us also define $\mathtt{PairFn}_k : \overline{k} \to \overline{k-1} \to \overline{k-1} \to \overline{0}$ by

$$\mathtt{PairFn}_k\ \mathtt{z}^k\ \mathtt{u}^{k-1}\ \mathtt{v}^{k-1} \equiv \mathtt{z\ (Pair}_{k-1}\ \mathtt{u\ v)}$$

Next, define the *level* of a type by

$$\begin{aligned} \mathrm{level}(\overline{0}) &= 0 \\ \mathrm{level}(\sigma \times \tau) &= \max(\mathrm{level}(\sigma), \mathrm{level}(\tau)) \\ \mathrm{level}(\sigma \to \tau) &= \max(1 + \mathrm{level}(\sigma), \mathrm{level}(\tau)) \end{aligned}$$

and for any type σ define $\overline{\sigma}$ to be the pure type $\overline{\mathrm{level}(\sigma)}$.

Proposition 6.3. (i) For any type σ and any $k \geq \mathrm{level}(\sigma)$, there exist defined constants

$$\mathtt{encode}^k_\sigma : \sigma \to \overline{k}, \qquad \mathtt{decode}^k_\sigma : \overline{k} \to \sigma, \qquad \mathtt{idem}^k_\sigma : \overline{k} \to \overline{k}$$

such that $\mathtt{idem}^k_\sigma$ is a regular defined constant, and (in any TTS T) for all $y \in T_\sigma$, $z \in T_{\overline{k}}$ we have

$$decode^k_\sigma \cdot (encode^k_\sigma \cdot y) \;=\; y, \qquad encode^k_\sigma \cdot (decode^k_\sigma \cdot z) \;=\; idem^k_\sigma \cdot z$$

(ii) Moreover, for any types σ, τ there is a regular defined constant

$$\mathtt{apply}_{\sigma\tau} : \overline{\sigma \to \tau} \to \overline{\sigma} \to \overline{\tau}$$

such that (in any TTS T) for all $h \in T_{\overline{\sigma\to\tau}}$, $z \in T_{\overline{\sigma}}$ we have

$$apply_{\sigma\tau} \cdot h \cdot z \;=\; encode^{\overline{\tau}}_{\tau} \cdot ((decode^{\overline{\sigma\to\tau}}_{\sigma\to\tau} \cdot h) \cdot (decode^{\overline{\sigma}}_{\sigma} \cdot z))$$

Proof. We define the constants $\mathtt{encode}^k_\sigma$, $\mathtt{decode}^k_\sigma$ and $\mathtt{idem}^k_\sigma$ simultaneously by induction on the structure of σ, with a subordinate induction on k. We take care to ensure that the definitions for $\mathtt{idem}^k_\sigma$ involve only regular defining equations. The induction cases are as follows:

— For the case $\sigma = \overline{0}$ and $k = 0$, take

$$\begin{aligned} \mathtt{encode}^0_0\ \mathtt{y}^0 &\equiv \mathtt{y} \\ \mathtt{decode}^0_0\ \mathtt{z}^0 &\equiv \mathtt{z} \\ \mathtt{idem}^0_0\ \mathtt{z}^0 &\equiv \mathtt{z} \end{aligned}$$

— Given $\mathtt{encode}^k_\sigma$, $\mathtt{decode}^k_\sigma$, $\mathtt{idem}^k_\sigma$, define

$$\begin{aligned} \mathtt{encode}^{k+1}_\sigma\ \mathtt{y}^\sigma &\equiv \mathtt{up}_k\ (\mathtt{encode}^k_\sigma\ \mathtt{y}) \\ \mathtt{decode}^{k+1}_\sigma\ \mathtt{z}^{k+1} &\equiv \mathtt{decode}^k_\sigma\ (\mathtt{down}_k\ \mathtt{z}) \\ \mathtt{idem}^{k+1}_\sigma\ \mathtt{z}^{k+1} &\equiv \mathtt{up}_k\ (\mathtt{idem}^k_\sigma\ (\mathtt{down}_k\ \mathtt{z})) \end{aligned}$$

— If $k = \mathrm{level}(\sigma \times \tau)$, define

$$\begin{aligned} \mathtt{encode}^k_{\sigma\times\tau}\ \mathtt{y}^{\sigma\times\tau} &\equiv \mathtt{Pair}_k\ (\mathtt{encode}^k_\sigma\ (\mathtt{fst}\ \mathtt{y}))\ (\mathtt{encode}^k_\tau\ (\mathtt{snd}\ \mathtt{y})) \\ \mathtt{decode}^k_{\sigma\times\tau}\ \mathtt{z}^k &\equiv \mathtt{pair}\ (\mathtt{decode}^k_\sigma\ (\mathtt{Fst}_k\ \mathtt{z}))\ (\mathtt{decode}^k_\tau\ (\mathtt{Snd}_k\ \mathtt{z})) \\ \mathtt{idem}^k_{\sigma\times\tau}\ \mathtt{z}^k &\equiv \mathtt{Pair}_k\ (\mathtt{idem}^k_\sigma\ (\mathtt{Fst}_k\ \mathtt{z}))\ (\mathtt{idem}^k_\tau\ (\mathtt{Snd}_k\ \mathtt{z})) \end{aligned}$$

— If $k = \mathrm{level}(\sigma \to \tau)$, define

$$\begin{aligned} \mathtt{encode}^k_{\sigma\to\tau}\ \mathtt{y}^{\sigma\to\tau}\ \mathtt{x}^{k-1} &\equiv \mathtt{encode}^k_\tau\ (\mathtt{y}\ (\mathtt{decode}^{k-1}_\sigma\ (\mathtt{Fst}_{k-1}\ \mathtt{x})))\ (\mathtt{Snd}_{k-1}\ \mathtt{x}) \\ \mathtt{decode}^k_{\sigma\to\tau}\ \mathtt{z}^k\ \mathtt{w}^\sigma &\equiv \mathtt{decode}^k_\tau\ (\mathtt{PairFn}_k\ \mathtt{z}\ (\mathtt{encode}^{k-1}_\sigma\ \mathtt{w})) \\ \mathtt{idem}^k_{\sigma\to\tau}\ \mathtt{z}^k\ \mathtt{x}^{k-1} &\equiv \mathtt{idem}^k_\tau\ (\mathtt{PairFn}_k\ \mathtt{z}\ (\mathtt{idem}^{k-1}_\sigma\ (\mathtt{Fst}_{k-1}\ \mathtt{x})))\ (\mathtt{Snd}_{k-1}\ \mathtt{x}) \end{aligned}$$

It is straightforward to show by induction that these constants have the required properties. At various points one must appeal to the extensionality of TTSs: that is, we show $f = g$ by showing $f \cdot x = g \cdot x$ for all x.

(ii) First, for any $k \geq l$ we may define regular constants $\mathtt{Up}^k_l : \overline{l} \to \overline{k}$, $\mathtt{Down}^k_l : \overline{k} \to \overline{l}$ as follows:

$$\begin{aligned} \mathtt{Up}^l_l\ \mathtt{u} &\equiv \mathtt{u} \\ \mathtt{Up}^{k+1}_l\ \mathtt{u} &\equiv \mathtt{up}_k\ (\mathtt{Up}^k_l\ \mathtt{u}) \\ \mathtt{Down}^l_l\ \mathtt{u} &\equiv \mathtt{u} \\ \mathtt{Down}^{k+1}_l\ \mathtt{u} &\equiv \mathtt{Down}^k_l\ (\mathtt{down}_k\ \mathtt{u}) \end{aligned}$$

Now suppose $k = \mathrm{level}(\sigma \to \tau)$, $l = \mathrm{level}(\sigma)$, $m = \mathrm{level}(\tau)$, and define

$$\mathtt{apply}_{\sigma\tau}\ \mathtt{h}^k\ \mathtt{z}^l \;\equiv\; \mathtt{idem}^m_\tau\ (\mathtt{Down}^k_m\ (\mathtt{PairFn}_k\ \mathtt{h}\ (\mathtt{Up}^k_l\ (\mathtt{idem}^l_\sigma\ \mathtt{z}))))$$

It is easy to verify that $\mathtt{apply}_{\sigma\tau}$ has the required property. □

The significance of the regular defined constants is that their interpretations may be easily determined just by looking at the pure type functionals. Given a TTS T and a regular definition environment $\Delta = (c_0^{\rho_0} \equiv e_0,\ \ldots\ , c_{s-1}^{\rho_{s-1}} \equiv e_{s-1})$, where $\rho_i = \sigma_{i0} \to \cdots \to \sigma_{i(r_i-1)} \to \overline{0}$, we may associate with each c_i a function $[\![\, c_i \,]\!] : T_{\sigma_{i0}} \times \cdots \times T_{\sigma_{i(r_i-1)}} \to \mathbb{N}$ in the following way. If c_i is one of the constants of Δ_0, we simply take $[\![\, c_i \,]\!]$ to be the corresponding primitive recursive function. Otherwise, suppose functions $[\![\, c_j \,]\!]$ have been defined for all $j < i$, and suppose that

$$c_i\ x_0^{\sigma_{i0}}\ \cdots x_{r-1}^{\sigma_{i(r-1)}}\ \equiv\ e$$

is a regular defining equation for c_i, where $r \leq r_i$. For any valuation ν assigning to each x_j $(0 \leq j < r)$ an element $\nu(x_j) \in T_{\sigma_{ij}}$, we may define a number or function $[\![\, e \,]\!]_\nu$ as follows, by induction on the structure of the regular term e:

— $[\![\, c_j \,]\!]_\nu = [\![\, c_j \,]\!]$
— $[\![\, x_j \,]\!]_\nu = \nu(x_j)$
— If $e' : \tau_0 \to \cdots \to \tau_{t-1} \to \overline{0}$ where $t > 0$ and the τ_j are pure, and $e'' : \tau_0$, define $[\![\, e'e'' \,]\!]_\nu$ by

$$[\![\, e'e'' \,]\!]_\nu(y_1, \ldots, y_{t-1})\ =\ [\![\, e' \,]\!]_\nu(![\![\, e'' \,]\!]_\nu, y_1, \ldots, y_{t-1})$$

where $![\![\, e'' \,]\!]_\nu$ denotes the element of T_{τ_0} corresponding to $[\![\, e'' \,]\!]_\nu$ (see below).

Finally, we define $[\![\, c_i \,]\!]$ itself by

$$[\![\, c_i \,]\!](y_0, \ldots, y_{r_i-1})\ =\ [\![\, e \,]\!]_\nu(y_r, \ldots, y_{r_i-1}), \quad \text{where } \nu(x_i) = y_i \text{ for } i < r.$$

Note, crucially, that the definition of $[\![\, c_i \,]\!]$ depends only on the pure part of T.

The above definitions are provisional in that we need to show that the required elements $![\![\, e'' \,]\!]_\nu$ always exist (when they do, they are unique by virtue of the extensionality of T). This and more is given by the following:

Proposition 6.4. Suppose $\Delta = (c_0^{\rho_0} \equiv e_0,\ \ldots\ , c_{s-1}^{\rho_{s-1}} \equiv e_{s-1})$ is a regular definition environment. Then for each $i < s$ the above definitions indeed yield a total function $[\![\, c_i \,]\!]$, and moreover for any $y_0 \in T_{\sigma_{i0}}, \ldots, y_{r_i-1} \in T_{\sigma i(r_i-1)}$ we have

$$[\![\, c_i \,]\!](y_0, \ldots, y_{r_i-1})\ =\ \ll c_i \gg \cdot\, y_0 \cdot \cdots \cdot y_{r_i-1}$$

Proof. By induction on i. If c_i is a constant of Δ_0 this is clear. Otherwise, assume the claim holds for all c_j where $j < i$, and suppose

$$c_i\ x_0^{\sigma_{i0}}\ \cdots x_{r-1}^{\sigma_{i(r-1)}}\ \equiv\ e$$

is a regular defining equation for c_i. Then it suffices to show that, for any environment ν with domain $x_0, \ldots, x_{r-1}$, the number or function $[\![\, e \,]\!]_\nu$ is well-defined, and $![\![\, e \,]\!]_\nu$ exists and is $\ll e \gg_\nu$. This is shown by an easy induction on the structure of e. □

It is now easy to see how the whole of a type structure T may be reconstructed up to isomorphism from its pure part. For any type σ, define

$$T_\sigma^\circ\ =\ \{z \in T_{\overline{\sigma}} \mid [\![\, \mathtt{idem}_\sigma \,]\!](z) = z\}$$

and define application operations $\cdot_{\sigma\tau} : T^\circ_{\sigma\to\tau} \times T^\circ_\sigma \to T^\circ_\tau$ by

$$h \cdot_{\sigma\tau} z \;=\; [\![\, \mathtt{apply}_{\sigma\tau} \,]\!](h, z)$$

Then from Propositions 6.3 and 6.4 it is clear that $T^\circ \cong T$: the appropriate mappings $T_\sigma \to T^\circ_\sigma$ are realized by the elements $encode_\sigma$, and their inverses by $decode_\sigma$. Hence, by Proposition 6.3(ii), the application operations defined above correspond under these isomorphisms to the original application operations in T. It follows that two total type structures are isomorphic if their pure parts are isomorphic.

We also need to consider the corresponding question for internal TTSs. It is convenient here to formulate the following "obvious" definition explicitly (part (ii) of the definition will also be useful in Sections 7 and 8).

Definition 6.5. Suppose $\mathcal{C}$ has finite products and a standard natural number object N, and suppose $\mathbf{T}, \mathbf{T}'$ are internal TTSs in $\mathcal{C}$. For each $l \in \mathbb{N}$, let app_l denote the application morphism $\mathbf{T}_{l+1} \times \mathbf{T}_l \to \mathbf{T}_0$ for $\mathbf{T}$ and app'_l the corresponding morphism for $\mathbf{T}'$.

(i) We say $\mathbf{T}, \mathbf{T}'$ are *isomorphic at the pure types* if there is an isomorphism $\imath_l : \mathbf{T}_l \to \mathbf{T}'_l$ for each l, such that $\imath_0$ is the composition of the standard isomorphisms $\mathbf{T}_0 \cong N \cong \mathbf{T}'_0$, and at each level l the isomorphisms commute with application:

$$\imath_0 \circ app_l \;=\; app'_l \circ (\imath_{l+1} \times \imath_l) \;:\; \mathbf{T}_{l+1} \times \mathbf{T}_l \to \mathbf{T}'_0$$

(ii) We say $\mathbf{T}, \mathbf{T}'$ are *isomorphic up to level* k if there is an isomorphism $\imath_l : \mathbf{T}_l \to \mathbf{T}'_l$ for each $l \leq k$ such that $\imath_0$ is standard and the above equation holds for each $l < k$.

Theorem 6.6. Suppose $\mathcal{C}$ is well-pointed, has finite products and a standard natural number object, and suppose $\mathbf{T}, \mathbf{T}'$ are internal TTSs in $\mathcal{C}$ which are isomorphic at the pure types. Then $\mathbf{T} \cong \mathbf{T}'$.

Proof. Assume $\mathbf{T}, \mathbf{T}'$ are as given, with isomorphisms $\imath_l : \mathbf{T}_l \to \mathbf{T}'_l$ as above. Let $T = \mathrm{Hom}(1, \mathbf{T})$, $T' = \mathrm{Hom}(1, \mathbf{T}')$. For any type σ, the term $\mathtt{idem}^{\overline{\sigma}}_{\sigma}$ defines elements $\ll\!\mathtt{idem}\!\gg \in T_{\overline{\sigma}\to\overline{\sigma}}$ and $\ll\!\mathtt{idem}\!\gg' \in T'_{\overline{\sigma}\to\overline{\sigma}}$. Since the pure parts of T, T' are isomorphic and $\mathtt{idem}^{\overline{\sigma}}_{\sigma}$ is regular, by Proposition 6.4 we have

$$\imath_{\overline{\sigma}} \circ (\ll\!\mathtt{idem}\!\gg \cdot z) \;=\; \ll\!\mathtt{idem}\!\gg' \cdot (\imath_{\overline{\sigma}} \circ z)$$

for all $z \in T_\sigma$. Now $\ll\!\mathtt{idem}\!\gg$ induces a morphism $[\![\, \mathtt{idem} \,]\!] : \mathbf{T}_{\overline{\sigma}} \to \mathbf{T}_{\overline{\sigma}}$ in $\mathcal{C}$ by means of the application morphism $\mathbf{T}_{\overline{\sigma}\to\overline{\sigma}} \times \mathbf{T}_{\overline{\sigma}} \to \mathbf{T}_{\overline{\sigma}}$, and similarly for $\mathbf{T}'$. Since $\mathcal{C}$ is well-pointed, we deduce that $\imath_{\overline{\sigma}} \circ [\![\, \mathtt{idem} \,]\!] = [\![\, \mathtt{idem} \,]\!]' \circ \imath_{\overline{\sigma}}$.

In the same way, the terms $\mathtt{encode}^{\overline{\sigma}}_{\sigma}, \mathtt{decode}^{\overline{\sigma}}_{\sigma}$ yield morphisms $[\![\, \mathtt{encode} \,]\!] : \mathbf{T}_\sigma \to \mathbf{T}_{\overline{\sigma}}$ and $[\![\, \mathtt{decode} \,]\!] : \mathbf{T}_{\overline{\sigma}} \to \mathbf{T}_\sigma$ respectively, and by Proposition 6.3(i) and the well-pointedness of $\mathcal{C}$ we may deduce that

$$[\![\, \mathtt{decode} \,]\!] \circ [\![\, \mathtt{encode} \,]\!] = id_{\mathbf{T}_\sigma}, \qquad [\![\, \mathtt{encode} \,]\!] \circ [\![\, \mathtt{decode} \,]\!] = [\![\, \mathtt{idem} \,]\!]$$

and similarly for $\mathbf{T}'$. Define $\imath_\sigma = [\![\, \mathtt{decode} \,]\!]' \circ [\![\, \mathtt{encode} \,]\!]$; then it is easy to check that $\imath_\sigma$ is an isomorphism $\mathbf{T}_\sigma \to \mathbf{T}'_\sigma$.

A similar argument now shows that the $\imath_\sigma$ commute with the application morphisms, using Proposition 6.3(ii) and the fact that $\mathtt{apply}_{\sigma\tau}$ is regular. □

In the cases of interest, of course, $\mathcal{C}$ will be some category $\mathbf{Asm}(B)$, so that internal TTSs in $\mathcal{C}$ correspond to realizations of TTSs over B. In such cases, the above result may be rephrased as follows: if T, T' are type structures with realizations γ, γ' over B such that γ, γ' are isomorphic at the pure types, then $\gamma \cong \gamma'$.

7. Proof for the continuous case

We are now ready to embark on the proofs of our main theorems. In this section we will prove the main theorems for C and $\mathsf{C}^{\mathit{eff}}$ (Theorems 5.13 and 5.14); the theorem for HEO will be proved in Section 8.

Throughout this section, we suppose A is a full continuous NR-TPCA, with a realization $\gamma : A \longrightarrow\!\triangleright K_2$ and associated elements $\mathbf{a}_{\sigma\tau}$ and $\mathbf{d}$ as in Definition 5.1, $\mathbf{h}$ as in Definition 5.5, and **norm** as in Theorem 5.13, such that conditions (AB) and (C) of Theorem 5.13 are satisfied. By the remarks at the end of Section 5.4, we may assume that $\mathbf{h}$ is normal.

To establish the main theorem for C, we wish to prove that $\gamma_*(\mathbf{EC}(A)) \cong \mathbf{C}$ in $\mathbf{Asm}(K_2)$. To do this, we will first formulate a lemma which asserts the existence of suitable "Normann programs" for all pure types, and will show that the lemma implies the desired isomorphism. We will then prove the lemma itself by an induction on the type level, and this is of course where the hard work comes.

To formulate the lemma, we make use of the model P. Recall from Section 5.3 the near-realization $\Vdash^{\mathsf{P}}_{\mathrm{B}}$ of $\mathsf{P}^{\mathit{eff}}$ over K_2 given by basic enumerations; this induces a genuine realization $\Vdash^{\mathsf{P}}_{\mathrm{B}} \circ \epsilon^{\mathsf{P}} : \mathsf{EC}(\mathsf{P}) \longrightarrow\!\triangleright K_2$. We write δ for the corresponding realization $\mathsf{C} \longrightarrow\!\triangleright K_2$; by Theorem 5.10(i) and the remarks at the end of Section 5.3, δ is isomorphic to $\Vdash^{\mathsf{C}}$.

The following terminology will be useful throughout the proof. Suppose $k \geq 1$, and suppose that $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{C}$ are indeed isomorphic up to level $k-1$, with isomorphisms $\imath_l : \gamma_*(\mathbf{EC}(A))_l \to \mathbf{C}_l$ for $l < k$ as in Definition 6.5. In this situation, we say an element $\dot{x} \in A_l$ *represents* some $x \in \mathsf{C}_l$ if $\dot{x} \in \mathsf{Tot}(A_l)$ and $[\dot{x}]$ corresponds to x under the isomorphism $\imath_l$. We may also say that $\dot{x} \in A_k$ *represents* $x \in \mathsf{C}_k$ if whenever $\dot{y} \in A_{k-1}$ represents $y \in \mathsf{C}_{k-1}$, $\dot{x} \cdot \dot{y}$ represents $x(y)$. (It follows from this that $x \in \mathsf{Tot}(A_k)$.) Likewise, for any $l \leq k$, we will say an element $\mathfrak{x} \in \mathsf{P}_l$ *represents* $x \in \mathsf{C}_l$ if $\mathfrak{x} \in \mathsf{Tot}(\mathsf{P}_l)$ and $[\mathfrak{x}]$ corresponds to x under the isomorphism $\mathsf{EC}(\mathsf{P}) \cong \mathsf{C}$. (Note that these uses of the term "represents" smoothly extend those of Section 2.2.) In general, if some symbol denotes an element of C, we will often use its dotted counterpart to denote an element of A that represents it, and its Gothic counterpart to denote an element of P that represents it.

As mentioned in Section 6.1, we may extend our language NRComb to a language NRComb$^+$ by adding a new constant $\mathtt{norm} : \overline{1} \to \overline{1}$. When considering interpretations of this language, we stipulate that $\ll\!\mathtt{norm}\!\gg = \mathbf{norm}$. We may now state our lemma as follows:

Lemma 7.1 (Main Lemma). Suppose $k \geq 1$, and suppose $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{C}$ are isomorphic up to level $k-1$. Then there is an NRComb$^+$-definable constant $\mathtt{nabla}_k : \overline{1} \to \overline{k}$ denoting an element $\nabla_k \in A_{\overline{1} \to \overline{k}}$ with the following property: whenever $\Phi \in \mathsf{C}_k$, $\nu \Vdash^{\delta}_k \Phi$ and $\dot{\nu} \in A_1$ represents ν, the element $\nabla_k \cdot \dot{\nu} \in A_k$ represents Φ.

An element ∇_k satisfying the above condition will be called a *Normann operator* for type $\overline{k}$ in A, and the program of NRComb$^+$ that defines it will be called a *Normann program.*

Theorem 7.2. Assuming Lemma 7.1, we have $\gamma_*(\mathbf{EC}(A)) \cong \mathbf{C}$.

Proof. By Theorem 6.6, it suffices to show that $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{C}$ are isomorphic at the pure types. We will construct suitable isomorphisms $\imath_k$ by induction on k.

For $k = 0$, we have an isomorphism $\imath_0 : \gamma_*(\mathbf{EC}(A))_0 \cong \mathbf{C}_0$ by virtue of the fact that $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{C}$ are both internal type structures. (This in turn depends on the fact that γ_* preserves the natural number object — see the remark following Proposition 5.2).

For the induction step, suppose $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{C}$ are isomorphic up to level $k-1$ via isomorphisms $\imath_l$ for $l < k$ as above, and let $\jmath_l = \imath_l^{-1}$ for each l. We will construct morphisms $\imath_k : \gamma_*(\mathbf{EC}(A))_k \to \mathbf{C}_k$ and $\jmath_k : \mathbf{C}_k \to \gamma_*(\mathbf{EC}(A))_k$ such that

$$\forall f \in \mathsf{EC}(A)_k,\ x \in \mathsf{EC}(A)_{k-1}.\quad \imath_k(f) \cdot \imath_{k-1}(x) = \imath_0(f \cdot x)$$
$$\forall f \in \mathsf{C}_k,\ x \in \mathsf{C}_{k-1}.\quad \jmath_k(f) \cdot \jmath_{k-1}(x) = \jmath_0(f \cdot x)$$

Since both $\mathsf{EC}(A)$ and C are extensional, this will imply that $\imath_k, \jmath_k$ are mutually inverse isomorphisms.

For $\imath_k$, we have a morphism

$$\imath_0 \circ \gamma_*(app_{k-1}) \circ (id \times \jmath_{k-1}) \ :\ \gamma_*(\mathbf{EC}(A))_k \times \mathbf{C}_{k-1} \longrightarrow \mathbf{C}_0$$

whose exponential transpose gives us a morphism

$$\imath_k \ :\ \gamma_*(\mathbf{EC}(A))_k \longrightarrow \mathbf{C}_0^{\mathbf{C}_{k-1}} = \mathbf{C}_k$$

and it is clear by construction that this morphism commutes with application.

We now have to construct $\jmath_k$. First, by Theorem 5.10(i) and the remarks at the end of Section 5.3, there is a realizer $\mathbf{r} \in K_2$ such that for any $\Phi \in \mathsf{C}_k$ and $g \Vdash_k^{\mathsf{C}} \Phi$ we have $\mathbf{r} \odot g \Vdash_k^{\delta} \Phi$ — that is, $\mathbf{r} \odot g$ is a basic enumeration ν of some total $\mathfrak{Phi} \in \mathsf{P}_k$ representing Φ. Next, using the realizer $\mathbf{h}$ from Definition 5.5, we have that $\mathbf{h} \odot (\mathbf{r} \odot g)$ is a γ-realizer for some $\dot{\nu} \in A_1$ representing ν. But now, by the main lemma for k, the element $\dot{\Phi} = \nabla_k \cdot \dot{\nu} \in A_k$ is total and represents Φ; moreover, if $nabla_k \in K_2$ is any γ-realizer for ∇_k then $\mathbf{a}_{1k} \odot nabla_k \odot (\mathbf{h} \odot (\mathbf{r} \odot g)) \Vdash_k^{\gamma} \dot{\Phi}$. Thus, by the combinatory completeness of K_2, given a realizer for $\Phi \in \mathsf{C}_k$ we may (within K_2) compute a realizer for $\Phi \in \gamma_*(\mathbf{EC}(A))_k$, so we have a morphism $\jmath_k : \mathbf{C}_k \to \gamma_*(\mathbf{EC}(A))_k$ commuting with application as required. □

A similar argument can also be used to establish our main theorem for $\mathsf{C}^{\mathit{eff}}$. To see this, suppose that A' is an effective sub-NR-TPCA of A with respect to γ in the sense of Definition 5.6, and that $\mathbf{norm} \in A'$. (It is still harmless to assume that $\mathbf{h}$ is normal when $\mathbf{h}$ is chosen from K_2^{eff}.) We wish to show that $\gamma_*(\mathbf{EC}(A; A')) \cong \mathbf{C}^{\mathit{eff}}$ in $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$; again, by Theorem 6.6 it suffices to show that these internal TTSs are isomorphic at the pure types.

Recall that $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$ has the same objects as $\mathbf{Asm}(K_2)$ but fewer morphisms, and that exponentials in $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$ are defined exactly as in $\mathbf{Asm}(K_2)$. Since the two categories have the same natural number object, it follows that each object $\mathbf{C}_k^{\mathit{eff}}$ in

$\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$ is identical to the object $\mathbf{C}_k$ in $\mathbf{Asm}(K_2)$, and moreover the application morphisms correspond. Likewise, the internal TTS $\mathbf{EC}(A; A')$ in $\mathbf{Asm}(A; A')$ is identical to $\mathbf{EC}(A)$ in $\mathbf{Asm}(A)$. Since the functor $\gamma_* : \mathbf{Asm}(A; A') \to \mathbf{Asm}(K_2, K_2^{\mathit{eff}})$ is simply the restriction of the functor $\gamma_* : \mathbf{Asm}(A) \to \mathbf{Asm}(K_2)$, it follows that the internal TTS $\gamma_*(\mathbf{EC}(A; A'))$ in $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$ is identical to $\gamma_*(\mathbf{EC}(A))$ in $\mathbf{Asm}(K_2)$. So in order to show that $\gamma_*(\mathbf{EC}(A; A')) \cong \mathbf{C}^{\mathit{eff}}$ in $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$, it only remains to show that the isomorphisms $\gamma_*(\mathbf{EC}(A))_k \cong \mathbf{C}_k$ constructed above exist in $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$, *i.e.*, are realizable in K_2.

An inspection of the proof of Theorem 7.2 readily confirms that this is indeed the case. The realizer $\mathbf{r}$ may be chosen in K_2^{eff} by Theorem 5.10(ii). Note also that $\nabla_k \in A'$, since the elements $\mathbf{k}, \mathbf{s}, \ldots$ from the definition of NR-TPCA are all in A', as is the element **norm**, and ∇_k may be obtained from these via application (see Definition 2.2). From Definition 5.6, it follows that a realizer nabla_k for ∇_k may be chosen in K_2^{eff}. The remaining details are trivial.

We have thus established:

Theorem 7.3. Assuming Lemma 7.1, we have $\gamma_*(\mathbf{EC}(A; A')) \cong \mathbf{C}^{\mathit{eff}}$ in $\mathbf{Asm}(K_2; K_2^{\mathit{eff}})$.

To complete the proof of both Theorem 5.13 and Theorem 5.14, it remains to prove Lemma 7.1 itself. The proof is by a grand induction on k. In Section 7.1 we will establish the main lemma for the cases $k = 1, 2$; in Sections 7.2 to 7.5 we will show that the main lemma holds for $k \geq 3$ assuming it holds for $k - 2$.

7.1. *Normann programs for types 1 and 2*

The existence of a Normann program for type 1 is straightforward. Let us write $f \in \mathsf{C}_1$ in place of Φ, and suppose $\nu \vdash_1^\delta f$. Then $\nu \in K_2$ basically enumerates some $\mathsf{f} \in \mathsf{Tot}(\mathsf{P}_1)$ which represents f. By the definition of $\Vdash_{\mathrm{B}}^{\mathsf{P}}$ in Section 5.3, each number in the range of ν is a type 1 basic code, *i.e.*, a number of the form $\langle a \mapsto \breve{n} \rangle$. We may therefore simulate f within A as follows: given $\widehat{m} \in A_0$, search ν for a basic code of the form $\langle \breve{m} \mapsto \breve{n} \rangle$ or $\langle \breve{\bot} \mapsto \breve{n} \rangle$, and return the corresponding numeral $\widehat{n}$.

Formally, we define derived constants of NRComb as follows, bearing in mind that $\breve{n}$ is simply $n + 1$ and $m \mapsto n$ is $\langle m, n \rangle$.

$$\begin{aligned}
\texttt{eq-or-zero x}^0\ \texttt{y}^0 &\equiv \texttt{or (eq x y) (eq x 0)} \\
\texttt{matches m}^0\ \texttt{b}^0 &\equiv \texttt{eq-or-zero (proj0 (proj0 b)) (suc m)} \\
\texttt{index-matches nu}^1\ \texttt{m}^0\ \texttt{j}^0 &\equiv \texttt{matches m (nu j)} \\
\texttt{nabla}_1\ \texttt{nu}^1\ \texttt{m}^0 &\equiv \texttt{pre (proj1 (proj0 (nu (min (index-matches nu m)))))}
\end{aligned}$$

Let $\nabla_1 \in A_{\overline{1} \to \overline{1}}$ be the element defined by $\texttt{nabla}_1$. To see that ∇_1 has the property required for Lemma 8.1, suppose f, ν and f are above, $\dot{\nu}$ represents ν, and $m \in \mathbb{N}$. Recalling the typographical conventions of Section 2.1, for any j we clearly have that $\mathit{index\text{-}matches} \cdot \dot{\nu} \cdot \widehat{m} \cdot \widehat{j}$ is tt if $\nu(j)$ is of the form $\langle \breve{m} \mapsto \breve{n} \rangle$ or $\langle \breve{\bot} \mapsto \breve{n} \rangle$, and ff otherwise. But since f is total, the range of ν must contain some basic code of one of these forms, and

for any such code we must have that $n = f(m)$. It follows easily that $\nabla_1 \cdot \dot{\nu} \cdot \widehat{m} = \widehat{f(m)}$ as required.

For the type 2 case we will write $F \in \mathsf{C}_2$ in place of Φ. Suppose $\nu \vdash_2^\delta F$; then ν basically enumerates some $\mathfrak{F} \in \mathsf{Tot}(\mathsf{P}_2)$ which represents F. By the definition of $\Vdash_{\mathrm{B}}^{\mathsf{P}}$ in Section 5.3, each number in the range of ν is a type 2 basic code, *i.e.*, a number of the form

$$c \;=\; \langle\langle a_0 \mapsto \breve{n}_0, \ldots, a_{r-1} \mapsto \breve{n}_{r-1}\rangle \mapsto \breve{p}\rangle$$

satisfying certain conditions. Such a code c represents the element $\zeta_2(c) \in \mathsf{P}_2$ defined by

$$\zeta_2(c)(\mathfrak{g}) \;=\; \begin{cases} p & \text{if } g(\zeta_0(a_i)) = n_i \text{ for each } i, \\ \bot & \text{otherwise} \end{cases}$$

The idea is that we can simulate F within A as follows: given some $g \in \mathsf{EC}(A)_1$, search through ν looking for a code

$$\langle\langle \breve{m}_0 \mapsto \breve{n}_0, \ldots, \breve{m}_{r-1} \mapsto \breve{n}_{r-1}\rangle \mapsto \breve{p}\rangle$$

such that $g(m_i) = n_i$ for each i. When such a code is found, we return the corresponding element $\widehat{p}$. This procedure gives us an element of A_2 which agrees with F on all total elements of A_1, and so witnesses the fact that $F \in \mathsf{EC}(A_2)$.

A minor problem arises from the fact that the codes enumerated by ν might involve type 1 codes of the form $\breve{\bot} \mapsto \breve{n}$, and it is nonsensical to test whether "$g(\bot) = n$" in A. However, the following lemma shows that for the purpose of simulating F we may simply ignore all such codes:

Lemma 7.4. Suppose ν basically enumerates a total $\mathfrak{F} \in \mathsf{P}_2$, and $g \in \mathsf{EC}(A)_1$. Then $\mathrm{range}(\nu)$ contains a code $\langle\langle \breve{m}_0 \mapsto \breve{n}_0, \ldots, \breve{m}_{r-1} \mapsto \breve{n}_{r-1}\rangle \mapsto \breve{p}\rangle$ such that $g(m_i) = n_i$ for each i.

Proof. Since g is total, there is an element $\mathfrak{g} \in \mathsf{Tot}(\mathsf{P}_1)$ defined by $\mathfrak{g}(n) = g(n)$, $\mathfrak{g}(\bot) = \bot$. Since $\mathfrak{F}$ is total, we must have $\mathfrak{F}(\mathfrak{g}) = p$ for some $p \in \mathbb{N}$. Hence there must be some basic compact $\mathfrak{c} = (\bigsqcup_{i<r'}(\mathfrak{a}_i \Rightarrow n_i')) \Rightarrow p \sqsubseteq \mathfrak{F}$ such that $\mathfrak{a}_i \Rightarrow n_i' \sqsubseteq \mathfrak{g}$ and $n_i' \neq \bot$ for each i. But since $\mathfrak{g}(\bot) = \bot$, we cannot have $\mathfrak{a}_i = \bot$ for any i; hence $\mathfrak{a}_i \in \mathbb{N}$ for each i. Moreover, $\mathfrak{c}$ must turn up somewhere in the enumeration ν, and it is easy to see that *any* code c for $\mathfrak{c}$ must have the required syntactic form. □

We may now give a formal definition of the Normann program for type 2:

```
agrees-with g¹ b⁰ ≡ and (not (eq (proj0 b) 0))
                        (eq (g (pre (proj0 b)))
                            (pre (proj1 b)))
agrees-with-all g¹ b-list⁰ ≡ all-in-list (agrees-with g) b-list
code-works-for g¹ c⁰ ≡ agrees-with-all g (proj0 (proj0 c))
index-works-for nu¹ g¹ j⁰ ≡ code-works-for g (nu j)
index-for nu¹ g¹ ≡ min (index-works-for nu g)
nabla₂ nu¹ g¹ ≡ pre (proj1 (proj0 (nu (index-for nu g))))
```

Let $\nabla_2 \in A_{\overline{1}\to\overline{2}}$ be the element defined by $\mathtt{nabla}_2$. To see that ∇_2 has the property required for Lemma 8.1, suppose F, $\mathfrak{F}$ and ν are as above, $\dot{\nu}$ represents ν, and $\dot{g} \in A_1$ represents an arbitrary $g \in \mathsf{EC}(A)_1 \cong \mathsf{C}_1$. We have to show that $\nabla_2 \cdot \dot{\nu} \cdot \dot{g} = \widehat{F(g)}$. It is easy to see that for any $c \in \mathbb{N}$, the element *code-works-for* $\cdot\, \dot{g} \cdot \widehat{c}$ is *tt* if c has the form $\langle\langle \breve{m}_0 \mapsto \breve{n}_0, \ldots, \breve{m}_{r-1} \mapsto \breve{n}_{r-1}\rangle \mapsto \breve{p}\rangle$ where $g(m_i) = n_i$ for each i, and *ff* otherwise. Hence *index-works-for* $\cdot\, \dot{\nu} \cdot \dot{g} \cdot \widehat{j}$ is *tt* or *ff* for each $j \in \mathbb{N}$, and by Lemma 7.4 there exists j such that *index-works-for* $\cdot\, \dot{\nu} \cdot \dot{g} \cdot \widehat{j} = tt$. So *index-for* $\cdot\, \dot{\nu} \cdot \dot{g} = \widehat{j}$ for some such j, where $\dot{\nu} \cdot \widehat{j} = \widehat{c}$ for some c of the above form. But now it is easy to see that $\nabla_2 \cdot \dot{\nu} \cdot \dot{g}$ is the corresponding numeral $\widehat{p}$, and since $(\bigsqcup_{i<r}(m_i \Rightarrow n_i)) \Rightarrow p \sqsubseteq \mathfrak{F}$, we have that $p = F(g)$.

7.2. *A notion of "partial element" in A_{k-2}*

We next set about establishing Lemma 7.1 for an arbitrary $k \geq 3$, assuming that it holds for $k-2$, and it is here that the real interest begins. Suppose $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{C}$ are isomorphic up to level $k-1$. Then they are certainly isomorphic up to $k-3$, so we have a Normann program $\mathtt{nabla}_{k-2} : \overline{1} \to \overline{k-2}$ with the property stated in Lemma 7.1. As a mild convenience, we may also assume that $\nabla_{k-2} \cdot x\!\downarrow$ for all $x \in A_1$ (in fact, the Normann operators that we construct will have this property anyway). Our task is now to construct a Normann program for type $\overline{k}$.

Suppose $\Phi \in \mathsf{C}_k$ and $\nu \Vdash^{\delta}_k \Phi$; then $\nu \in K_2$ basically enumerates some $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}_k)$ which represents Φ. The codes enumerated by ν each have the form

$$\langle\langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1}\rangle \mapsto \breve{q}\rangle$$

where each $b_i \in \mathrm{dom}\ \zeta_{k-2}$. Our Normann program will ultimately have to operate on enumerations ν of this kind, in which type $k-2$ elements are represented by ζ_{k-2} codes. However, a crucial auxiliary role will be played by another way of representing type $k-2$ elements, which we now describe.

First, suppose θ basically enumerates some total element of P_{k-2}, and define $\mu^\theta : \mathbb{N} \to \mathbb{N}$ by $\mu^\theta(i) = \theta(i) + 1$. We call μ^θ the *modified basic enumeration* obtained from θ. We also say $\mu : \mathbb{N} \to \mathbb{N}$ is a modified basic enumeration for $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ if $\mu = \mu^\theta$ for some basic enumeration θ of $\mathfrak{h}$.

Next, if $b = \langle b_0, \ldots, b_{t-1}\rangle$ is a ζ_{k-2}-code for some compact element of P_{k-2}, define a function $\mu^b : \mathbb{N} \to \mathbb{N}$ by

$$\mu^b(i) \;=\; \begin{cases} \langle b_i\rangle + 1 & \text{if } i < t, \\ 0 & \text{if } i \geq t \end{cases}$$

We call μ^b the modified basic enumeration obtained from b. We say μ is a modified basic enumeration for $\mathfrak{b} \in \mathsf{P}^{comp}_{k-2}$ if $\mu = \mu^b$ for some ζ_{k-2}-code b for $\mathfrak{b}$.

Clearly, there is a program $\mathtt{mod\text{-}enum} : \overline{0} \to \overline{1}$ such that if $b \in \mathrm{dom}\ \zeta_{k-2}$ then *mod-enum* $\cdot\, \widehat{b}$ represents μ^b. Conversely, there is a program $\mathtt{code} : \overline{1} \to \overline{0}$ such that if $\dot{\mu}^b \in A_1$ represents μ^b then *code* $\cdot\, \dot{\mu}^b = \widehat{b}$.

The purpose of these notions is to allow us to express total elements as limits of compact elements at the level of intensional representations. The problem we are seeking to address is the following: given an arbitrary $\dot{G} \in \mathsf{Tot}(A_{k-1})$ representing $G \in \mathsf{C}_{k-1}$,

we know how $\dot{G}$ behaves on total elements of A_{k-2}, but we have no idea *a priori* how $\dot{G}$ behaves on non-total elements. (This contrasts somewhat with the situation in (Normann 2000), where one at least knew that $\dot{G}$ was monotone and continuous.) The idea is that we are going to use modified basic enumerations of compact elements to define certain "potentially partial elements" of A_{k-2}. Since the total elements arise as limits of these, we can use continuity in K_2 to gain some purchase on how $\dot{G}$ behaves on such partial elements.

In order to construct these potentially partial elements, we will write a program `interpret` which, given a modified basic enumeration μ^b for a compact element $\mathfrak{b} \in \mathsf{P}_{k-2}$, yields an element $\dot{b} \in A_{k-2}$ which we can loosely regard as a kind of counterpart to the partial element $\mathfrak{b}$.[‡‡]

The idea is to attempt to "extend" μ^b to obtain an enumeration for a total element which we may then pass to ∇_{k-2}. This extension will be constructed with the help of a program[§§]

$$\texttt{critical-index} \ : \ \overline{k-1} \to \overline{1} \to \overline{0}$$

which we shall leave unspecified for the time being (it will be defined in Section 7.5 below). The arguments passed to this program will be the element $\dot{G}$ in question, and a representation of a certain function $\lambda : \mathbb{N} \to \mathbb{N}$ whose role will be explained later. This program will also have the property that *critical-index*$\cdot \dot{G}\!\downarrow$ for all $\dot{G} \in A_{k-1}$.

First, it is convenient to define a thunked version of `critical-index` with a further dummy argument.

$$\texttt{critical-index-thunk}\ \texttt{G}^{k-1}\ \texttt{lam}^{1}\ \ \texttt{dummy}^{0} \quad \equiv \quad \lceil \texttt{critical-index G} \rceil\ \texttt{lam}$$

The reason for the protected subexpression on the right hand side (here and in several later definitions) will emerge in the proof of Lemma 7.11. Note that *critical-index-thunk*$\cdot \dot{G} \cdot \dot{\lambda}\!\downarrow$ for all $\dot{G} \in A_{k-1}$ and $\dot{\lambda} \in A_1$ by Proposition 2.7.

The next piece of machinery gives us a way of "extending compact elements to total elements", whilst leaving total elements as they are. More precisely, if μ is a modified basic enumeration for some compact or total element $\mathfrak{h}_0$ of type $k-2$, we may try to extend it to an enumeration of some total element $\mathfrak{h} \sqsupseteq \mathfrak{h}_0$, and thence obtain an actual element of A_{k-2}. This extension will be constructed using one of the standard enumerations $\xi_{k-2,c}$ from Theorem 4.9. The particular index c that is used will be computed by evaluating a given thunked index. The value of the thunk is then "filtered" to ensure that the resulting total element $\mathfrak{x}_c$ is indeed above $\mathfrak{h}_0$.

In the following definitions, we suppose the defined constant $\texttt{below} : \overline{0}^{(2)} \to \overline{0}$ uniformly

[‡‡] Actually, towards the end of the proof it will emerge that, in the cases of interest, our "potentially partial elements" are total after all! For motivational purposes, however, it seems best to ignore this for the time being and loosely think of the element $\dot{b}$ as being genuinely partial.

[§§] Strictly speaking, the constant `critical-index` and most of the other derived constants we shall introduce should be annotated with k, since we require separate versions of these programs at each level of the induction. We shall omit these annotations except in the crucial case of $\texttt{nabla}_k$ and $\texttt{nabla}_{k-2}$.

represents the function

$$(c, c') \;\mapsto\; \begin{cases} \mathit{tt} & \text{if } c, c' \in \mathrm{dom}\ \zeta_{k-2} \text{ and } \zeta_{k-2}(c) \sqsubseteq \zeta_{k-2}(c'), \\ \mathit{ff} & \text{otherwise} \end{cases}$$

and $\mathtt{xi} : \overline{0}^{(2)} \to \overline{0}$ uniformly represents the total computable function $(c, i) \mapsto \xi_{k-2,c}(i)$ as in Theorem 4.9.

$$\begin{aligned}
\mathtt{filter}\ \mathtt{b}^0\ \mathtt{c}^0 &\equiv \mathtt{if}\ (\mathtt{below}\ \mathtt{b}\ \mathtt{c})\ \mathtt{c}\ \mathtt{b} \\
\mathtt{filter\text{-}enum}\ \mathtt{mu}^1\ \mathtt{thunk}^1\ \mathtt{i}^0 &\equiv \mathtt{xi}\ (\mathtt{filter}\ (\mathtt{code}\ \mathtt{mu})\ (\mathtt{thunk}\ \mathtt{0}))\ \mathtt{i} \\
\mathtt{pre\text{-}fn}\ \mathtt{mu}^1\ \mathtt{i}^0 &\equiv \mathtt{pre}\ (\mathtt{mu}\ \mathtt{i}) \\
\mathtt{extend\text{-}enum}\ \mathtt{mu}^1\ \mathtt{thunk}^1\ \mathtt{i}^0 &\equiv \mathtt{if}_1\ (\mathtt{not}\ (\mathtt{eq}\ (\mathtt{mu}\ \mathtt{i})\ \mathtt{0})) \\
&\qquad\quad (\mathtt{pre\text{-}fn}\ \mathtt{mu}) \\
&\qquad\quad (\mathtt{filter\text{-}enum}\ \mathtt{mu}\ \mathtt{thunk}) \\
&\qquad\quad \mathtt{i} \\
\mathtt{partial\text{-}elt}\ \mathtt{mu}^1\ \mathtt{thunk}^1 &\equiv \mathtt{nabla}_{k-2}\ (\mathtt{extend\text{-}enum}\ \mathtt{mu}\ \mathtt{thunk})
\end{aligned}$$

(Note the use of thunking in the definition of `extend-enum`: the final `i` serves as a potential rightmost argument for both branches of the conditional.) The essential properties of `partial-elt` are as follows. Here and for the remainder of Section 7, $U_{\mathfrak{b}}$ denotes the neighbourhood $U^{\mathsf{C}}_{\mathfrak{b}}$ as defined in Section 4.2.

Proposition 7.5. (i) If $\dot{\mu}$ represents a modified basic enumeration of some $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing $h \in \mathsf{C}_{k-2}$, then for any *thunk* $\in A_1$, *partial-elt* $\cdot\, \dot{\mu} \cdot$ *thunk* is an element of $\mathsf{Tot}(A_{k-2})$ representing h (whether or not *thunk* $\cdot\, \widehat{0}\downarrow$).

(ii) If $\dot{\mu}$ represents a modified basic enumeration of some $\mathfrak{b} \in \mathsf{P}^{comp}_{k-2}$, and *thunk* $\cdot\, \widehat{0} = \widehat{c}$, then *partial-elt* $\cdot\, \dot{\mu} \cdot$ *thunk* is an element of $\mathsf{Tot}(A_{k-2})$ representing some $h \in U_{\mathfrak{b}}$.

(iii) If $\dot{\mu}$ represents a modified basic enumeration of some $\mathfrak{b} \in \mathsf{P}^{comp}_{k-2}$, and $\mathfrak{b}'$ is any compact element above $\mathfrak{b}$ so that $U_{\mathfrak{b}'} \subseteq U_{\mathfrak{b}}$, then there exists $c \in \mathbb{N}$ such that, whenever *thunk* $\cdot\, \widehat{0} = \widehat{c}$, *partial-elt* $\cdot\, \dot{\mu} \cdot$ *thunk* represents some $h \in U_{\mathfrak{b}'}$.

Proof. (i) If $\dot{\mu}$ represents a modified basic enumeration μ of a total element $\mathfrak{h}$ representing h, then *not* $\cdot$ (*eq* $\cdot$ ($\dot{\mu} \cdot \widehat{i}$) $\cdot\, \widehat{0}$) $=$ *tt* for all i. Moreover, for any *thunk* $\in A_1$ we have *filter-enum* $\cdot\, \dot{\mu} \cdot$ *thunk*$\downarrow$, and so *extend-enum* $\cdot\, \dot{\mu} \cdot$ *thunk* $\cdot\, \widehat{i} = \widehat{\mu(i) \dot{-} 1}$ for all i. That is, *extend-enum* $\cdot\, \dot{\mu} \cdot$ *thunk* represents an ordinary basic enumeration of $\mathfrak{h}$, and so by the hypothesis for $\mathtt{nabla}_{k-2}$, *partial-elt* $\cdot\, \dot{\mu} \cdot$ *thunk* represents h.

(ii) If $\dot{\mu}$ represents a modified basic enumeration μ of a compact element $\mathfrak{b}$, and *thunk* $\cdot\, \widehat{0}$ evaluates to some numeral, then clearly *filter* $\cdot$ (*code* $\cdot\, \dot{\mu}$) $\cdot$ (*thunk* $\cdot\, \widehat{0}$) evaluates to a numeral $\widehat{b'}$ such that $\mathfrak{b} \sqsubseteq \mathfrak{b}'$ where $\mathfrak{b}' = \zeta_{k-2}(b')$. Hence, by Theorem 4.9, *filter-enum* $\cdot\, \dot{\mu} \cdot$ *thunk* represents a basic enumeration $\xi_{k-2,b'}$ of some $\mathfrak{x} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing some $h \in U_{\mathfrak{b}'} \subseteq U_{\mathfrak{b}}$. Let t be the least number such that $\mu(t) = 0$; then by condition (3) in Theorem 4.9, we have $\bigsqcup_{i \geq t} \zeta_{k-2}(\xi_{k-2,b'}(i)) = \mathfrak{x}$. But also if $i < t$ then $\zeta_{k-2}(\mu(i) \dot{-} 1) \sqsubseteq \mathfrak{b} \sqsubseteq \mathfrak{x}$, and it follows that *extend-enum* $\cdot\, \dot{\mu} \cdot$ *thunk* also represents a basic enumeration of $\mathfrak{x}$. The result follows by the hypothesis on $\mathtt{nabla}_{k-2}$.

(iii) Given any $\mathfrak{b}' \in \mathsf{P}^{comp}_{k-2}$ above $\mathfrak{b}$, let c be any code for $\mathfrak{b}'$. Then *filter* $\cdot\, \widehat{b} \cdot \widehat{c} = \widehat{c}$, and

as in (ii) it follows that if $\mathit{thunk} \cdot \widehat{0} = \widehat{c}$ then $\mathit{partial\text{-}elt} \cdot \dot{\mu} \cdot \mathit{thunk}$ represents an element $h \in U_{\mathfrak{b}'}$. □

Combining the pieces, the following gives us a way of interpreting a modified basic enumeration μ^b as a potentially partial element in A_{k-2}:

$$\begin{aligned}\texttt{interpret}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{mu}^1 \quad &\equiv \quad \texttt{partial-elt mu} \\ &\qquad (\lceil\texttt{critical-index-thunk G}\rceil\ \texttt{lam})\end{aligned}$$

Since we do not yet know whether $\mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda}\downarrow$, there is at present no guarantee that $\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \dot{\mu}^b$ is a total element of A_{k-2}. However, this element is at least defined, since $\mathit{critical\text{-}index\text{-}thunk} \cdot \dot{G} \cdot \dot{\lambda}\downarrow$ and $\nabla_{k-2} \cdot x\downarrow$ for all x.

Given a basic code $\langle b \mapsto \check{p}\rangle$ for type $k-1$, the following programs can be regarded as testing whether $\dot{G}$ satisfies the condition corresponding to $b \mapsto \check{p}$. The strange packaging of the arguments for `G-value` and `satisfies` is a mild convenience in order to facilitate the proof of Proposition 7.6 below.

$$\begin{aligned}\texttt{G-value}\ \texttt{G}^{k-1}\ \texttt{lam-mu}^{\overline{1}\times\overline{1}} \quad &\equiv \quad \texttt{G}\ (\lceil\texttt{interpret G}\rceil \\ &\qquad (\texttt{fst lam-mu})\ (\texttt{snd lam-mu})) \\ \texttt{satisfies}\ \texttt{G}^{k-1}\ \texttt{p}^0\ \texttt{lam-mu}^{\overline{1}\times\overline{1}} \quad &\equiv \quad \texttt{eq}\ (\lceil\texttt{G-value G}\rceil\ \texttt{lam-mu})\ \texttt{p} \\ \texttt{satisfies-code}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{b}^0\ \texttt{p}^0 \quad &\equiv \quad \lceil\texttt{satisfies G}\rceil\ \texttt{p} \\ &\qquad (\texttt{pair}\ (\texttt{norm lam}) \\ &\qquad\qquad (\texttt{norm}\ (\texttt{mod-enum b})))\end{aligned}$$

Clearly, if $\dot{\mu}$ represents a modified basic enumeration of some total $\mathfrak{h}$ representing $h \in \mathsf{C}_{k-2}$, then $\dot{h} = \mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \dot{\mu}$ also represents h, so $\mathit{satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot (\mathit{pair} \cdot \dot{\lambda} \cdot \dot{\mu})$ will be *tt* if $G(h) = p$, and *ff* otherwise.

7.3. *The "graph" of a total type $k-1$ element*

Our argument now proceeds roughly as follows. If indeed we are in the scenario just mentioned and $G(h) = p$, then by the continuity of K_2, it must be possible to replace $\dot{\mu}$ by some $\dot{\mu}^b$ such that $\mathit{satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot (\mathit{pair} \cdot \dot{\lambda} \cdot \dot{\mu}^b)$ is still *tt*. The code b then gives us some kind of "modulus of continuity" for $\dot{G}$ at $\dot{h}$, with the pair (b, p) serving as a kind of "partial element" for $\dot{G}$. By doing this for every μ corresponding to a total element, we may obtain a set of such partial elements that comprise a complete graph for some element $\mathfrak{G} \in \mathsf{P}_{k-1}$ representing G. We may think of $\mathfrak{G}$ as embodying the implicit modulus information that we have extracted from (the γ-realization of) the element $\dot{G} \in A_{k-1}$.

Once we have a *bona fide* element of $\mathsf{Tot}(\mathsf{P}_{k-1})$, our total functional $\mathfrak{P}\mathfrak{h}\mathfrak{i} \in \mathsf{Tot}(\mathsf{P}_k)$ will surely know what to do with it. Indeed, by scanning the given enumeration ν of Φ and comparing its elements with those of the graph of $\mathfrak{G}$, we will eventually discover what $\mathfrak{P}\mathfrak{h}\mathfrak{i}(\mathfrak{G})$ is, and this will tell us what $\nabla_k \cdot \dot{\nu} \cdot \dot{G}$ ought to be.

We now give the detailed argument leading to the construction of $\mathfrak{G}$ (parametrically in a function $\lambda : \mathbb{N} \to \mathbb{N}$). The following crucial proposition shows how the behaviour of $\dot{G}$ on any $\dot{h} \in \mathsf{Tot}(A_{k-2})$ is in some sense determined by a finite amount of information

about $\dot{h}$. This is the only point in the proof at which condition (AB) of Theorem 5.13 is used. We also show that, in a certain sense, only a finite amount of information about the function λ is involved. As a useful piece of notation, for any $\lambda \in \mathbb{N}^{\mathbb{N}}$ and $z \in \mathbb{N}$ we may define the corresponding *Baire neighbourhood*:

$$\mathcal{B}(\lambda, z) \;=\; \{\lambda' \in \mathbb{N}^{\mathbb{N}} \mid \forall i < z.\, \lambda'(i) = \lambda(i)\}$$

Proposition 7.6. Suppose given $\dot{G} \in \mathsf{Tot}(A_{k-1})$ representing $G \in \mathsf{C}_{k-1}$, $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing $h \in \mathsf{C}_{k-2}$, and $\lambda \in \mathbb{N}^{\mathbb{N}}$, and suppose $G(h) = p$. Then there exist $b, z \in \mathbb{N}$ with $\zeta_{k-2}(b) \sqsubseteq \mathfrak{h}$ such that

(1) for any $\dot{\lambda}' \in \mathsf{Tot}(A_1)$ representing some $\lambda' \in \mathcal{B}(\lambda, z)$, we have

$$\mathit{satisfies\text{-}code} \cdot \dot{G} \cdot \dot{\lambda}' \cdot \widehat{b} \cdot \widehat{p} \;=\; \mathit{tt}$$

(2) for all $h' \in O_{k-2,b}$ we have $G(h') = p$.

Proof. Let θ be any basic enumeration of $\mathfrak{h}$. Recall that $\mu^{\theta\sharp}$ is the element of A_1 realized by $\mathbf{h} \odot \mu^{\theta}$ and that $\mathbf{norm} \cdot \mu^{\theta\sharp} = \mu^{\theta\sharp}$; likewise for $\lambda^\sharp$. By Proposition 7.5(i), $\mathit{partial\text{-}elt}\cdot\mu^{\theta\sharp}\cdot\mathit{thunk}$ represents h for any $\mathit{thunk} \in A_1$, so $\dot{h} = \mathit{interpret}\cdot\dot{G}\cdot\lambda^\sharp\cdot\mu^{\theta\sharp}$ represents h. Now $\dot{G}\cdot\dot{h} = \widehat{G(h)} = \widehat{p}$, so $(\mathit{satisfies}\cdot\dot{G}\cdot\widehat{p})\cdot(\mathbf{pair}\cdot\dot{\lambda}\cdot\dot{\mu}^{\theta}) = \mathit{tt}$. But $\mathbf{pair}\cdot\lambda^\sharp\cdot\mu^{\theta\sharp}$ has a realizer $\alpha = \mathit{pair}' \odot (\mathbf{h} \odot \lambda) \odot (\mathbf{h} \odot \mu^{\theta})$, where pair' is a realizer for the evident pairing operation. So by condition (AB) of Theorem 5.13 with $\sigma = \overline{1} \times \overline{1}$, recalling that $\mathit{tt} = \widehat{0}$, there is some open set $V \subseteq K_2$ with $\alpha \in V$, such that whenever $\alpha' \in V$ realizes some $\mathit{lam\text{-}mu}' \in A_{\overline{1}\times\overline{1}}$ we have $\mathit{satisfies}\cdot\dot{G}\cdot\widehat{p}\cdot\mathit{lam\text{-}mu}' = \mathit{tt}$. But the mapping $(\lambda', \mu') \mapsto \mathit{pair}' \odot (\mathbf{h}\odot\lambda')\odot(\mathbf{h}\odot\mu')$ is a total continuous function $K_2 \times K_2 \to K_2$, so there exist $z, t \in \mathbb{N}$ such that for all $\lambda' \in \mathcal{B}(\lambda, z)$ and $\mu' \in \mathcal{B}(\mu^{\theta}, t)$ we have $\mathit{pair}' \odot (\mathbf{h} \odot \lambda') \odot (\mathbf{h} \odot \mu') \in V$. Let b be the code $\langle \mathrm{proj}\ 0\,(\theta(0)), \ldots, \mathrm{proj}\ 0\,(\theta(t-1))\rangle$, so that $\mu^b(i) = \mu^{\theta}(i)$ for $i < t$ and $\mu^b(i) = 0$ for $i \geq t$. Note that $\zeta_{k-2}(b) \sqsubseteq \mathfrak{h}$.

To see that b and z satisfy condition (1), suppose $\dot{\lambda}' \in A_1$ represents some $\lambda' \in \mathcal{B}(\lambda, z)$. Then $\mathbf{norm} \cdot \dot{\lambda}' = \lambda'^\sharp$, and likewise $\mathbf{norm} \cdot (\mathit{mod\text{-}enum} \cdot \widehat{b}) = \mu^{b\sharp}$. Hence

$$\mathit{satisfies\text{-}code} \cdot \dot{G} \cdot \dot{\lambda}' \cdot \widehat{b} \cdot \widehat{p} \;=\; \mathit{satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot (\mathit{pair} \cdot \lambda'^\sharp \cdot \mu^{b\sharp}) \;=\; \mathit{tt},$$

since $\mathit{pair} \cdot \lambda'^\sharp \cdot \mu^{b\sharp}$ has the realizer $\mathit{pair}' \odot (\mathbf{h} \odot \lambda') \odot (\mathbf{h} \odot \mu^b) \in V$.

Finally, we show that b satisfies condition (2). Given any $h' \in O_{k-2,b}$, by Proposition 4.10(ii) we may find $\mathfrak{h}' \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing h' with $\zeta_{k-2}(b) \sqsubseteq \mathfrak{h}'$, so we may choose an enumeration θ' for $\mathfrak{h}'$ such that $\theta' \in \mathcal{B}(\theta, t)$. But now $\mathit{pair}' \odot (\mathbf{h}\odot\lambda)\odot(\mathbf{h}\odot\mu^{\theta'}) \in V$, so $\mathit{satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot (\mathit{pair} \cdot \lambda^\sharp \cdot \mu^{\theta'\sharp}) \;=\; \mathit{tt}$. But $\mathit{interpret} \cdot \dot{G} \cdot \lambda^\sharp \cdot \mu^{\theta'\sharp}$ represents h' by Proposition 7.5(i), so $G \cdot (\mathit{interpret} \cdot \dot{G} \cdot \lambda^\sharp \cdot \mu^{\theta'\sharp}) = \widehat{G(h')}$, whence $G(h') = p$. □

We will say that a pair (b, p) is *satisfactory* for $\dot{G}$ relative to λ if $b \in \mathrm{dom}\ \zeta_{k-2}$ and there exists $z \in \mathbb{N}$ such that b, p, z satisfy conditions (1) and (2) of the above proposition. We write $S_\lambda(\dot{G})$ for the set of all satisfactory pairs for $\dot{G}$ relative to λ. The next proposition shows how this set gives rise to a complete "graph" for G.

Proposition 7.7. Suppose given $\dot{G} \in \mathsf{Tot}(A_{k-1})$ representing $G \in \mathsf{C}_{k-1}$, and $\lambda \in \mathbb{N}^{\mathbb{N}}$. Then the supremum $\bigsqcup\{\zeta_{k-1}(b \mapsto \breve{p}) \mid (b, p) \in S_\lambda(\dot{G})\}$ exists and is an element $\mathfrak{G}_\lambda$ of $\mathsf{Tot}(\mathsf{P}_{k-1})$ representing G.

Proof. Let $S = S_\lambda(\dot{G})$. We first show that if $(b,p),(b',p') \in S$ then the elements $\zeta_{k-1}(b \mapsto \breve{p})$, $\zeta_{k-1}(b' \mapsto \breve{p'})$ are consistent in P_{k-1}. Suppose $\zeta_{k-2}(b), \zeta_{k-2}(b')$ are consistent in P_{k-2}. Then by Theorem 4.9 there exists $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ such that $\zeta_{k-2}(b), \zeta_{k-2}(b') \sqsubseteq \mathfrak{h}$. So if $h \in \mathsf{C}_{k-2}$ is the element represented by $\mathfrak{h}$ then $h \in O_{k-2,b}$ by Proposition 4.10(ii); hence $G(h) = p$ since (b,p) is satisfactory for $\dot{G}$ relative to λ. But similarly $G(h) = p'$, so $p = p'$.

We may therefore define $\mathfrak{G}_\lambda = \bigsqcup\{\zeta_{k-1}(b \mapsto \breve{p}) \mid (b,p) \in S\}$. To see that $\mathfrak{G}_\lambda$ is total and represents G, suppose $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ represents $h \in \mathsf{C}_{k-2}$. By Proposition 7.6, there exists a pair $(b,p) \in S$ such that $\zeta_{k-2}(b) \sqsubseteq \mathfrak{h}$ and $p = G(h)$. But $\zeta_{k-1}(b \mapsto \breve{p}) \sqsubseteq \mathfrak{G}$, so $\mathfrak{G}(\mathfrak{h}) = p = G(h)$. □

7.4. *Satisfactory representations for type k objects*

We now explain the role of the parameter λ. Recall that we are given a basic enumeration ν, and wish to simulate the corresponding Φ within A. To this end, we first need to convert ν into an alternative representation of Φ involving the satisfactory pairs discussed above. A small piece of notation will be useful: given $\lambda : \mathbb{N} \to \mathbb{N}$ and $j \in \mathbb{N}$, define $\lambda^j : \mathbb{N} \to \mathbb{N}$ by $\lambda^j(i) = \lambda(j+i)$.

Definition 7.8. Suppose $\Phi \in \mathsf{C}_k$. A *satisfactory representation* of Φ is a total function $\lambda : \mathbb{N} \to \mathbb{N}$ such that

(1) for all j, the number $\lambda(j)$ is either 0 or of the form $2(c \mapsto \breve{q})+1$ or $2(c \mapsto \breve{q})+2$, where $c \in \operatorname{dom} \zeta_{k-1}$ and $q \in \mathbb{N}$;
(2) whenever $\lambda(j) = 2(c \mapsto \breve{q}) + 1$, there exists $j' > j$ such that $\lambda(j'') = 0$ whenever $j < j'' < j'$, and $\lambda(j') = 2(c \mapsto \breve{q}) + 2$;
(3) whenever $\lambda(j) = 2(c \mapsto \breve{q}) + 1$, the basic ζ_k-code $\langle c \mapsto q \rangle$ is consistent with Φ (that is, $\Phi \in O_{k,\,\langle c \mapsto q\rangle}$);
(4) for any $\dot{G} \in \mathsf{Tot}(A_{k-1})$, there exists j_0 such that $\lambda(j_0)$ has the form

$$2(\langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1}\rangle \mapsto \breve{q}) + 1$$

where each pair (b_i, p_i) is satisfactory for $\dot{G}$ relative to λ^{j_0+1}.

The numbers appearing in the range of λ will be called *tokens*. The idea is that given $\dot{G}$, our program will search through the tokens enumerated by λ, looking for a token $\lambda(j) = 2(c \mapsto \breve{q}) + 1$ satisfying condition (4) above, which can be used to determine the value of $\Phi(G)$. When this token is tested using *satisfies-code*, the remaining tail λ^{j+1} is passed as a parameter so that later tokens may be (recursively) tested. The purpose of the zero tokens is to arrange that λ^{j+1} is (for certain j) indistinguishable from the constant zero function as far as *satisfies-code* is concerned, in that all the interesting information in λ^{j+1} is "hidden out of sight". The purpose of the tokens $2(c \mapsto \breve{q})+2$ is to allow the current token $x = \lambda(j)$ to be retrieved from λ^{j+1} — this avoids having to pass x as a separate parameter, which would cause problems for our notion of satisfactory pair.

It is not too hard to construct satisfactory representations:

Proposition 7.9. Every functional $\Phi \in \mathsf{C}_k$ has a satisfactory representation $\lambda : \mathbb{N} \to \mathbb{N}$. Moreover, a satisfactory representation of Φ is computable from a basic enumeration ν for Φ. That is, there is a type 2 partial computable function satis-rep : $\mathbb{N}^{\mathbb{N}} \times \mathbb{N} \rightharpoonup \mathbb{N}$ such that for any $\Phi \in \mathsf{C}_k$ and any $\nu \Vdash^{\delta}_k \Phi$, the function satis-rep$(\nu, -)$ is total and is a satisfactory representation of Φ.

Proof. Given $\nu \Vdash^{\delta}_k \Phi$, we may first construct an enumeration ν' of all possible basic codes for all elements enumerated by ν:

$$\text{range}(\nu') \;=\; \{d \text{ a basic } \zeta_k\text{-code} \mid \exists j.\ \zeta_k(\nu(j)) = \zeta_k(d)\}$$

Clearly, such a ν' may be computed uniformly from ν, since the relation "$\zeta_k(d) = \zeta_k(d')$" is decidable. Next, from ν' we may construct a function $\lambda : \mathbb{N} \to \mathbb{N}$ with the following properties:

— For each j, $\lambda(j)$ is either 0 or of the form $2e+1$ or $2e+2$ where $\langle e \rangle \in \text{range}(\nu')$.

— For each $\langle e \rangle \in \text{range}(\nu')$, the value $2e+1$ occurs as $\lambda(j)$ for infinitely many j.

— For each $\langle e \rangle \in \text{range}(\nu')$ and $z > 0$, the zth occurrence of $2e+1$ in λ is immediately followed in λ by a sequence of z zeros and then the token $2e+2$. That is, if j_z is the zth natural number such that $\lambda(j_z) = 2e+1$, then $\lambda(j'') = 0$ for all j'' with $j_z < j'' \leq j_z + z$, and $\lambda(j_z + z + 1) = 2e+2$.

Again, it is easy to see how a suitable function λ may be computed uniformly from ν', and this establishes the computability requirement in the proposition.

It remains to show that λ is indeed a satisfactory representation of Φ. Conditions (1), (2) and (3) are clear, since every basic type k code has the form $\langle c \mapsto \breve{q} \rangle$ where $c \in \text{dom}\ \zeta_{k-1}$, and if $\zeta_k(\langle c \mapsto \breve{q} \rangle) = \zeta_k(\nu(j))$ for some j then $\Phi \in U_{\zeta_k(\langle c \mapsto \breve{q} \rangle)}$, so $\Phi \in O_{k,\, \langle c \mapsto q \rangle}$ by Proposition 4.10(ii).

For condition (4), suppose $\dot{G} \in \mathsf{Tot}(A_{k-1})$ represents $G \in \mathsf{C}_{k-1}$. Let λ_0 denote the constant zero function, and let $\mathfrak{G}_{\lambda_0} \in \mathsf{Tot}(\mathsf{P}_{k-1})$ be the element obtained from $\dot{G}$ and λ_0 as in Proposition 7.7. Then $\mathfrak{G}_{\lambda_0}$ represents G and $\bigsqcup_j \zeta_k(\nu(j))$ represents Φ, so we must have $\zeta_k(\nu(j))(\mathfrak{G}_{\lambda_0}) = \Phi(G)$ for some j. Suppose $\nu(j) = \langle c \mapsto \breve{q} \rangle$ where $q = \Phi(G)$; then $\zeta_{k-1}(c) \sqsubseteq \mathfrak{G}_{\lambda_0}$, so there is some finite list of pairs $(b_0, p_0), \ldots, (b_{r-1}, p_{r-1}) \in S_{\lambda_0}(\dot{G})$ such that $\zeta_{k-1}(c) \sqsubseteq \bigsqcup_{i<r} \zeta_{k-1}(\langle b_i \mapsto \breve{p_i} \rangle)$. Let $c' = \langle b_0 \mapsto \breve{p_0}, \ldots, b_{r-1} \mapsto \breve{p}_{r-1} \rangle$; then $\zeta_k(c' \mapsto \breve{q}) \sqsubseteq \zeta_k(c \mapsto \breve{q}) = \zeta_k(\nu(j))$, so the compact element $\zeta_k(c' \mapsto \breve{q})$ must appear somewhere in the enumeration ν. Hence the code $\langle c' \mapsto \breve{q} \rangle$ itself appears somewhere in ν'. Now for each $i < r$ we may choose z_i so that b_i, p_i, z_i satisfy conditions (1) and (2) of Proposition 7.6 with $\lambda = \lambda_0$. Let z be the maximum of the z_i, and let j_0 be the index of the zth occurrence of $2(c' \mapsto \breve{q}) + 1$ in λ. Then $\mathcal{B}(\lambda^{j_0+1}, z) = \mathcal{B}(\lambda_0, z)$, and so z witnesses the satisfactoriness of each (b_i, p_i) relative to λ^{j_0+1}. □

7.5. *The Normann search algorithm*

Having shown how to compute a satisfactory representation λ of Φ, we now start to assemble the main program for computing $\Phi(G)$ itself. From here on, our proof closely follows the argument in (Normann 2000).

Some simple pieces of machinery in NRComb will be useful. We assume `plus`, `half`,

`even` are defined constants representing the evident arithmetical functions and predicate. First, a program for the operation mapping λ to λ^j:

$$\begin{aligned} \texttt{shift}'\ \texttt{j}^0\ \texttt{lam}^1\ \texttt{i}^0 &\equiv \texttt{lam}\ (\texttt{plus j i}) \\ \texttt{shift}\ \texttt{j}^0\ \texttt{lam}^1 &\equiv \texttt{norm}\ (\texttt{shift}'\ \texttt{j lam}) \end{aligned}$$

Given a token $x = 2(c \mapsto \breve{q}) + 1$, the following programs extract the values of c and q:

$$\begin{aligned} \texttt{left-code}\ \texttt{x}^0 &\equiv \texttt{proj0}\ (\texttt{half}\ (\texttt{pre x})) \\ \texttt{result}\ \texttt{x}^0 &\equiv \texttt{pre}\ (\texttt{proj1}\ (\texttt{half}\ (\texttt{pre x}))) \end{aligned}$$

We write left-code, result for the corresponding ordinary functions $\mathbb{N} \to \mathbb{N}$.

The following program `lookup` is used to retrieve a token $x = 2(c \mapsto \breve{q}) + 1$ from some "tail" λ^{j+1} where $\lambda(j) = x$, using the fact that the first non-zero token appearing in λ^{j+1} is $x + 1$.

$$\begin{aligned} \texttt{positive}\ \texttt{lam}^1\ \texttt{j}^0 &\equiv \texttt{not}\ (\texttt{eq}\ (\texttt{lam j})\ \texttt{0}) \\ \texttt{lookup lam} &\equiv \texttt{pre}\ (\texttt{lam}\ (\texttt{min}\ (\texttt{positive lam}))) \end{aligned}$$

Given $\dot{\nu}$ and $\dot{G}$, our main program will search through the tokens enumerated by the corresponding satisfactory representation λ, looking for a token $x = 2(c \mapsto \breve{q}) + 1$ such that c is "satisfied" by $\dot{G}$. Formally, the program `search`, defined as follows, returns the position index of the first such token in λ.

$$\begin{aligned} \texttt{sat}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{t} &\equiv \lceil \texttt{satisfies-code G} \rceil\ \texttt{lam} \\ &\qquad (\texttt{proj0 t})\ (\texttt{pre}\ (\texttt{proj1 t})) \\ \texttt{sat-all}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{x} &\equiv \texttt{if}\ (\texttt{even x})\ \texttt{ff} \\ &\qquad (\texttt{all-in-list}\ (\lceil \texttt{sat G} \rceil\ \texttt{lam})\ (\texttt{left-code x})) \\ \texttt{sat-all-pos}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{j}^0 &\equiv \texttt{sat-all G}\ (\texttt{shift}\ (\texttt{suc j})\ \texttt{lam})\ (\texttt{lam j}) \\ \texttt{search}\ \texttt{lam}^1\ \texttt{G}^{k-1} &\equiv \texttt{min}\ (\texttt{sat-all-pos G lam}) \end{aligned}$$

Let $\texttt{satis-rep} : \overline{1} \to \overline{0} \to \overline{0}$ be an NRComb program that uniformly represents the type 2 partial computable function satis-rep of Proposition 7.9. Then the Normann program itself for type k may be defined as follows:

$$\begin{aligned} \texttt{nabla}'\ \texttt{lam}^1\ \texttt{G}^{k-1} &\equiv \texttt{result}\ (\texttt{lam}\ (\texttt{search lam G})) \\ \texttt{nabla nu}^1\ \texttt{G} &\equiv \texttt{nabla}'\ (\texttt{satis-rep nu})\ \texttt{G} \end{aligned}$$

There remain two problems to be solved. Firstly, because of the uncertain effect of applying $\dot{G}$ to "partial elements" in A, it is not immediately guaranteed that the relevant tests $sat \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{t}$ will always yield a recognizable truth value (that is, an element of $\mathcal{T}$ or $\mathcal{F}$). If the result of such a test is either undefined or some funny value in A_0, the whole computation is likely to be derailed. Secondly, even if $sat \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{t}$ yields "true", where $t = b \mapsto \breve{p}$, this does not in itself imply that $G(h') = p$ for all $h' \in O_b$ — in other words, it could be that condition (1) of Proposition 7.6 holds, but not condition (2). There is therefore a danger that $sat\text{-}all \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x}$ might yield "true" even though $G \notin O_c$ (where $x = 2(c \mapsto \breve{q}) + 1$), so that the computation as a whole might return a false value q.

It is precisely these problems that are addressed by Normann's argument. Recall from Section 7.2 that the definition of `satisfies` involved a term

$$\texttt{critical-index} \ : \ \overline{k-1} \to \overline{1} \to \overline{0}$$

which was left unspecified. We will define this program below in such a way that, in all relevant instances, the following properties will hold:

(1) The element $\dot{h} = \mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \dot{\mu}$ is a total element of A_1, where $\dot{\mu}$ represents some μ^b. This will ensure that $\dot{G} \cdot \dot{h}$ is a genuine numeral, and hence that $\mathit{sat} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{t}$ is a genuine truth value, where $t = b \mapsto \breve{p}$.

(2) If $\lambda(j) = 2(c \mapsto \breve{q}) + 1$ where $c = \langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1} \rangle$, and for each $i < r$ we have $\dot{G} \cdot (\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \dot{\mu}) = \widehat{p_i}$ whenever $\dot{\mu}$ represents μ^{b_i}, then $\Phi(G) = q$. Together with (1), this will ensure that whenever $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x}$ comes out as true, the corresponding value q that is returned is indeed the correct result.

However, in the situation of (2), it need not always be true that $G \in O_c$. Thus, in a certain sense, the Normann program may sometimes return "the right answer for the wrong reason".

Let us attempt an informal description of how $\widehat{m} = \mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda}$ is to be computed. We refer the reader to (Normann 2000) for further help.

First, recall from Section 7.2 the purpose for which m is to be used. We wish to extend the modified enumeration μ^b arising from some $b \in \mathrm{dom}\ \zeta_{k-2}$ to an enumeration of one of our standard total elements $\mathfrak{r}_m$, and thence to obtain a total element $\dot{h}_{b,m} \in A_{k-2}$ representing some $h_{b,m} \in \mathsf{C}_{k-2}$. This will in turn be passed to G, and the result compared with some p. If this test succeeds for all pairs (b_i, p_i) in question (*i.e.*, all the pairs appearing within some token $x = 2(c \mapsto \breve{q}) + 1$ where $c = \langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1} \rangle$), the whole test $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x}$ will succeed.

The computation of m using *critical-index* consists of two interleaved searches. The purpose of the first search is to try to find a "worst index" m which will cause the above test to fail if any will. For simplicity, first suppose the list c contains just one element $b \mapsto \breve{p}$. Then *critical-index* will search for an index m such that $\zeta(b) \sqsubseteq \zeta(m)$ but $G(h_{b,m}) \neq p$. If such an m is found, it is returned as the critical index — this will force the test $\mathit{satisfies\text{-}code} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{b} \cdot \widehat{p}$ to fail, so the *sat-all* test as a whole will fail. Moreover, this is what we want to happen: the "counterexample" $h_{b,m}$ shows that $G \notin O_c$, so the token x is not applicable to G anyway.

In the general case where c contains several elements $b_i \mapsto \breve{p}_i$, *critical-index* will search for an index m such that for at least one i we have $G(h_{b_i,m}) \neq p_i$. This means that the corresponding test $\mathit{satisfies\text{-}code} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{b_i} \cdot \widehat{p_i}$ will fail, so again the *sat-all* test as a whole will fail. (Each of the *satisfies-code* tests will separately discover the same value of m, and it is enough that this m causes at least one test to fail.) Again, the relevant function $h_{b_i,m}$ provides a counterexample showing that $2(c \mapsto \breve{q}) + 1$ is not applicable to G.

Of course, this search will be fruitless if no such m exists. However, we interleave this "first search" with another search, which looks further down λ for another token which can be used to settle the value of $\Phi(G)$ (a tricky recursion is involved here). If this "second search" reveals that the value of $\Phi(G)$ is something other than q, then this

too (indirectly) provides evidence that $G \notin O_c$. In this case, we resume the first search for an index m that provides a counterexample, now with the confidence that we will eventually find one.

On the other hand, if the second search reveals that $\Phi(G) = q$, we still do not know whether $G \in O_c$, but this scarcely interests us any more, since we know that q is the final answer we wish to return. In this case, we may as well choose some value of m that forces the *sat-all* condition to come out true, so that q will be returned immediately. (This is the scenario in which we might sometimes return "the right answer for the wrong reason".) Such an m is easy to supply, in view of the presumed failure of the first search so far.

The magic of this algorithm is that it always terminates. The reason has to do with condition (3) of Definition 7.8, which guarantees that λ eventually produces a "satisfactory" token which gives rise to the right answer for the right reason (albeit thanks to the miracle of continuity rather than our programming skills). All paths through the recursive computation tree will eventually hit this marvellous token, and the resulting knowledge will then "propagate back" through the entire tree ensuring that all stages of the computation are correct.

Our computation will (recursively) search through a given satisfactory enumeration λ until a relevant satisfactory token is found. However, rather than using a position index j to keep track of our place within λ, we will repeatedly "shift" the function λ as we pass it around so that all the tokens we have already examined are discarded. At each stage, the shifted element $\dot{\lambda}^1$ is renormalized using `norm`. This is done so that we do not pollute the argument we are going to pass to G with traces of the implicit value of j — thus, the value of $\mathit{satisfies\text{-}code} \cdot \dot{G} \cdot \dot{\lambda}$ will be quite independent of how far we have progressed through the original satisfactory representation. (In the effective setting we will meet the same problem, but will address it in a completely different way — see Section 8.3.)

We now embark on the formal definition of the program `critical-index`.

First, we need an enumeration of the dense set of standard total elements to be used in the first search. Using the machinery introduced in Section 7.2, we may define

$$\texttt{std-total}\ \mathtt{b}^0\ \mathtt{m}^0 \quad \equiv \quad \texttt{partial-elt}\ (\texttt{mod-enum}\ \mathtt{b})\ (\mathtt{k}\ \mathtt{m})$$

By Proposition 7.5, if $b \in \mathrm{dom}\ \zeta_{k-2}$ then the elements $\mathit{std\text{-}total} \cdot \widehat{b} \cdot \widehat{m}$ ($m \in \mathbb{N}$) are all total in A_{k-2} and correspond to a dense family of elements of $O_b \subseteq \mathsf{C}_{k-2}$. We call these elements the *standard total extensions* of b.

Given a code $t = b \mapsto \breve{p}$, we may test whether G agrees with t on the standard total extension of b with index m:

$$\begin{aligned}\texttt{std-satisfies}\ \mathtt{G}^{k-1}\ \mathtt{m}^0\ \mathtt{t}^0 \quad \equiv \quad & \texttt{eq}\ (\mathtt{G}\ (\texttt{std-total}\ (\texttt{proj0}\ \mathtt{t})\ \mathtt{m}))\\ & \qquad (\texttt{pre}\ (\texttt{proj1}\ \mathtt{t}))\end{aligned}$$

Next, given $x = 2(c \mapsto \breve{q}) + 1$ where $c = \langle t_0, \ldots, t_{r-1} \rangle$, we may test whether the index m provides a witness that G does not satisfy c (*i.e.* that there is some t_i which G does not satisfy):

$$\begin{aligned}\texttt{refutes}\ \mathtt{G}^{k-1}\ \mathtt{x}^0\ \mathtt{m}^0 \quad \equiv \quad & \texttt{and}\ (\texttt{not}\ (\texttt{even}\ \mathtt{x}))\\ & \qquad (\texttt{not}\ (\texttt{all-in-list}\ (\texttt{std-satisfies}\ \mathtt{G}\ \mathtt{m})\end{aligned}$$

(`left-code x`)))

This gives us all the machinery we need for the "first search". The following facts will be useful:

Proposition 7.10. Suppose $x = 2(c \mapsto \breve{q}) + 1$ where $c \in \mathrm{dom}\ \zeta_{k-1}$, and $\dot{G} \in \mathsf{Tot}(A_{k-1})$ represents $G \in \mathsf{C}_{k-1}$. Then

(i) for any $m \in \mathbb{N}$, $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x} \cdot \widehat{m} \in \mathcal{T} \cup \mathcal{F}$;

(ii) if $G \notin O_{k-1,c}$, there exists m such that $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x} \cdot \widehat{m} \in \mathcal{T}$.

Proof. Suppose $c = \langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1} \rangle$. For (i), note that for any m, each element $\mathit{extend\text{-}enum} \cdot (\mathit{mod\text{-}enum} \cdot \widehat{b_i}) \cdot (\mathbf{k} \cdot \widehat{m})$ represents a basic enumeration of some total element $\mathfrak{x}_{m'}$. So by the main induction hypothesis, $\mathit{std\text{-}total} \cdot \widehat{b_i} \cdot \widehat{m}$ represents this total element, whence $\dot{G} \cdot (\mathit{std\text{-}total} \cdot \widehat{b_i} \cdot m)$ is a numeral. It follows easily that $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x} \cdot \widehat{m} \in \mathcal{T} \cup \mathcal{F}$.

For (ii), if $G \notin O_{k-1,c}$ then $G \notin O_{k-1,\langle b_i \mapsto \breve{p}_i \rangle}$ for some i. That is, there exists some $h \in O_{k-2,b_i}$ such that $G(h) \neq p_i$. By Proposition 4.10(ii) there exists $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing h with $\mathfrak{h} \sqsupseteq \zeta_{k-2}(b_i)$, and by considering a representative $\mathfrak{G} \in \mathsf{Tot}(\mathsf{P}_{k-1})$ of G, we see that there must be some $\mathfrak{b} \sqsupseteq \zeta_{k-2}(b_i)$ such that $G(h') = G(h) \neq p_i$ for all $h' \in U_{\mathfrak{b}}$. Let m be a ζ_{k-2}-code for $\mathfrak{b}$; then clearly $\mathit{std\text{-}total} \cdot \widehat{b_i} \cdot \widehat{m}$ represents $\mathfrak{x}_m \in U_{\mathfrak{b}}$, so $\dot{G} \cdot (\mathit{std\text{-}total} \cdot \widehat{b_i} \cdot m)$ is a numeral other than $\widehat{p_i}$. It follows that $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x} \cdot \widehat{m} \in \mathcal{T}$. □

For the "second search", the program `sat-all` defined earlier provides most of the machinery, since it tests whether a number m is a position index for some token in λ that can be used to settle the value of $\Phi(G)$. It therefore remains to combine the two searches in the manner described above.

From this point onwards, our definitions involve a crucial use of recursion. For readability, we first present the definitions in a "circular" fashion, making use of the program `sat-all` which is supposed to have been constructed from `critical-index`. We will then explain how this circular definition may be understood as a well-founded definition involving the combinator $\mathtt{y}_2$.

In the next few definitions, the parameter `lam` should be understood to refer to the portion of the satisfactory representation that comes *after* the current token.

The following program (which invokes `sat-all`) is used for the "second search" — it tests whether m is a position index for some token in λ, occurring after the current position j_0, which can be used to settle the value of $\Phi(G)$. (The definition is virtually the same as that of `sat-all-pos`; we employ both names for the sake of analogy with the effective case.)

$$\mathtt{settles}\ \mathtt{G}^{k-1}\ \mathtt{lam}^1\ \mathtt{m}^0 \quad \equiv \quad \lceil \mathtt{sat\text{-}all\ G} \rceil\ (\mathtt{shift}\ (\mathtt{suc\ m})\ \mathtt{lam})\ (\mathtt{lam\ m})$$

We achieve the interleaving of the first and second searches with the following, which applies both tests to the same indices m:

$$\begin{aligned} \mathtt{test\text{-}index}\ \mathtt{G}^{k-1}\ \mathtt{lam}^1\ \mathtt{x}^0\ \mathtt{m}^0 \quad &\equiv \quad \mathtt{or}\ (\mathtt{refutes\ G\ x\ m}) \\ &\qquad\quad (\lceil \mathtt{settles\ G} \rceil\ \mathtt{lam\ m}) \\ \mathtt{pre\text{-}critical\text{-}index}\ \mathtt{G}^{k-1}\ \mathtt{lam}^1\ \mathtt{x}^0 \quad &\equiv \quad \mathtt{min}\ (\lceil \mathtt{test\text{-}index\ G} \rceil\ \mathtt{lam\ x}) \end{aligned}$$

If the index m discovered by the above is a refuter for G and x, we return m as the critical index. Otherwise, the token $\lambda(m)$ settles the value of $\Phi(G)$. If this value agrees with the value q given by the current element x, we wish to return a critical index that is *not* a refuter for G and x, so we may again return m. If the value disagrees with q, we restart the first search — this will eventually find a refuter which we then return.

$$\begin{aligned}\texttt{test-pre-critical}\ \texttt{G}^{k-1}\ \texttt{lam}^{1}\ \texttt{x}^{0}\ \texttt{m}^{0} \quad \equiv \quad & \texttt{if}\ (\texttt{or}\ (\texttt{refutes G x m}) \\ & \qquad (\texttt{and}\ (\texttt{not}\ (\texttt{even}\ (\texttt{lam m}))) \\ & \qquad\quad (\texttt{eq}\ (\texttt{result x}) \\ & \qquad\qquad (\texttt{result}\ (\texttt{lam m}))))) \\ & \qquad \texttt{m}\ \ (\texttt{min}\ (\texttt{refutes G x}))\end{aligned}$$

We may now put the pieces together. Note that `critical-index` does not take the token x in question as an explicit parameter, but this token has to be retrieved from λ using the program `lookup` defined at the beginning of this section.

$$\begin{aligned}\texttt{find-critical}\ \texttt{G}^{k-1}\ \texttt{lam}^{1}\ \texttt{x}^{0} \quad &\equiv \quad \texttt{test-pre-critical G lam x} \\ &\qquad (\lceil\texttt{pre-critical-index G}\rceil\ \texttt{lam x}) \\ \texttt{critical-index*}\ \texttt{G}^{k-1}\ \texttt{lam} \quad &\equiv \quad \lceil\texttt{find-critical G}\rceil\ \texttt{lam}\ (\texttt{lookup lam})\end{aligned}$$

The asterisk signals that this is not yet the official definition of `critical-index` itself. We next show how to eliminate the circularity from the above definition.

We will introduce variants of the above definitions which carry around an auxiliary parameter $\texttt{aux} : \overline{2}$. This should be thought of as a candidate for `critical-index G`, which we are going to define recursively. (Note that $\dot{G}$ stays fixed throughout the computation, and does not actively participate in the recursion.) We start by defining

$$\texttt{critical-index}'\ \texttt{G}^{k-1}\ \texttt{aux}^{2} \quad \equiv \quad \texttt{aux}$$

Next, recall the definition of `critical-index-thunk` from Section 7.2:

$$\texttt{critical-index-thunk}\ \texttt{G}^{k-1}\ \texttt{lam}^{1}\ \texttt{dummy}^{0} \quad \equiv \quad \lceil\texttt{critical-index G}\rceil\ \texttt{lam}$$

We adapt this definition so as to obtain a corresponding primed variant:

$$\texttt{critical-index-thunk}'\ \texttt{G}^{k-1}\ \texttt{aux}^{2}\ \texttt{lam}^{1}\ \texttt{dummy}^{0} \quad \equiv \quad \lceil\texttt{critical-index}'\ \texttt{G aux}\rceil\ \texttt{lam}$$

We apply a similar procedure to all the other constants dependent on `critical-index`, in the order of definition. In general, suppose $P : \overline{k-1} \to \tau$ is a constant which we defined earlier by means of an equation

$$P\ \texttt{G}^{k-1}\ x_0\ \cdots x_{r-1} \quad \equiv \quad E[\ \lceil Q\ \texttt{G}\rceil\ ]$$

where $r > 0$, $E[-]$ is a syntactic expression context which abstracts the (unique) occurrence of a protected subexpression on the right hand side of the definition of P, and $Q : \overline{k-1} \to \sigma$ is some constant. Suppose moreover that we have already defined a variant $Q' : \overline{k-1} \to \overline{2} \to \sigma$. We may then define the variant $P' : \overline{k-1} \to \overline{2} \to \tau$ by means of the equation

$$P\ \texttt{G}^{k-1}\ \texttt{aux}^{2}\ x_0\ \cdots x_{r-1} \quad \equiv \quad E[\ \lceil Q'\ \texttt{G aux}\rceil\ ]$$

We apply this procedure in turn to the definitions of the following constants:

`critical-index-thunk`, `interpret`, `G-value`, `satisfies`, `satisfies-code`, `sat`, `sat-all`, `settles`, `test-index`, `pre-critical-index`, `find-critical`

Note that for each of these constants P, the definition given earlier has $\mathtt{G}^{k-1}$ as the first formal parameter and at least one other parameter x_0, and moreover involves only one occurrence of a previously defined constant Q dependent on `critical-index`, which appears as part of a protected subexpression $\lceil Q\ \mathtt{G}\rceil$. Finally, we may define

$$\begin{aligned}\mathtt{critical\text{-}index''}\ \mathtt{G}^{k-1}\ \mathtt{aux}^2\ \mathtt{lam}^1 &\equiv \mathtt{find\text{-}critical'}\ \mathtt{G}\ \mathtt{aux}\\ &\qquad\qquad \mathtt{lam}\ (\mathtt{lookup}\ \mathtt{lam})\\ \mathtt{critical\text{-}index}\ \mathtt{G}^{k-1} &\equiv \mathtt{y}_2\ (\mathtt{critical\text{-}index''}\ \mathtt{G})\end{aligned}$$

Note at once that $\mathit{critical\text{-}index}\cdot\dot{G}\downarrow$ for all $\dot{G}\in A_{k-1}$, as specified in Section 7.2.

Now that we have a well-founded definition of `critical-index`, the definitions given earlier for all the non-primed constants listed above (and hence our definition of $\mathtt{nabla}_k$ itself) may be allowed to stand as the official definitions of these constants. We may now take $\nabla_k\in A_{\overline{1}\to\overline{k}}$ to be the element defined by $\mathtt{nabla}_k$.

7.6. *Correctness of the Normann program*

It remains to show that ∇_k indeed possesses the property required for Lemma 7.1. To this end, we first establish a useful property of the program `find-critical`. The proof makes use of the protected subexpressions appearing in our code, and indeed the purpose of these protected expressions is to ensure this property.

Lemma 7.11. For any $\dot{G}\in A_{k-1}$ and $\dot{\lambda}\in A_1$ we have

$$\mathit{critical\text{-}index}\cdot\dot{G}\cdot\dot{\lambda}\ \succeq\ \mathit{find\text{-}critical}\cdot\dot{G}\cdot\dot{\lambda}\cdot(\mathit{lookup}\cdot\dot{\lambda})$$

Proof. Suppose $\dot{G}\in A_{k-1}$, and let $\mathit{aux}=\mathbf{y}_2\cdot(\mathit{critical\text{-}index''}\cdot\dot{G})$; this is defined since $\mathbf{y}_2\cdot f\downarrow$ for any f, and indeed we have $\mathit{critical\text{-}index}\cdot\dot{G}=\mathit{aux}$. So by the definition of `critical-index'`, we have

$$\mathit{critical\text{-}index'}\cdot\dot{G}\cdot\mathit{aux}\ =\ \mathit{critical\text{-}index}\cdot\dot{G}$$

We now claim that for each of the constants P listed above, we have

$$\ll P'\gg\cdot\dot{G}\cdot\mathit{aux}\ =\ \ll P\gg\cdot\dot{G}$$

(that is, both sides are defined and they are equal). Suppose the definition of P is of the form

$$P\ \mathtt{G}^{k-1}\ x_0\ \cdots x_{r-1}\ \equiv\ E[\,\lceil Q\ \mathtt{G}\rceil\,]$$

as above, and assume we already know that $\ll Q'\gg\cdot\dot{G}\cdot\mathit{aux}=\ll Q\gg\cdot\dot{G}$. If P' is defined as above, then we have

$$\begin{aligned}\ll P'\gg\cdot\dot{G}\cdot\mathit{aux} &\succeq \ll(\lambda^* x_0\ldots x_{r-1}.\,E[\mathtt{z}])\gg_{\mathtt{z}\mapsto\ll Q'\gg\cdot\dot{G}\cdot\mathit{aux}}\\ &\simeq \ll(\lambda^* x_0\ldots x_{r-1}.\,E[\mathtt{z}])\gg_{\mathtt{z}\mapsto\ll Q\gg\cdot\dot{G}}\\ &\preceq \ll P\gg\cdot\dot{G}\end{aligned}$$

where $\mathbf{z}$ is a fresh variable. But $\ll(\lambda^* x_0 \ldots x_{r-1}.\, E[\mathbf{z}])\gg_{\mathbf{z}\mapsto \ll Q\gg\cdot\dot{G}}$ is defined since $r > 0$, $\ll Q\gg \cdot \dot{G}\!\downarrow$ and $E[\mathbf{z}]$ contains no protected subexpressions, so $\ll P'\gg \cdot \dot{G} \cdot \mathit{aux} = \ll P\gg \cdot \dot{G}$ as required.

In particular, we conclude that $\mathit{find\text{-}critical}' \cdot \dot{G} \cdot \mathit{aux} = \mathit{find\text{-}critical} \cdot \dot{G}$.

Now suppose $\dot{\lambda} \in A_1$. Using the axiom $\mathbf{y}_2 \cdot f \cdot x \succeq f \cdot (\mathbf{y}_2 \cdot f) \cdot x$, we have that

$$\begin{aligned}
\mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda} &\succeq \mathbf{y}_2 \cdot (\mathit{critical\text{-}index}'' \cdot \dot{G}) \cdot \dot{\lambda} \\
&\succeq \mathit{critical\text{-}index}'' \cdot \dot{G} \cdot \mathit{aux} \cdot \dot{\lambda} \\
&\succeq \mathit{find\text{-}critical}' \cdot \dot{G} \cdot \mathit{aux} \cdot \dot{\lambda} \cdot (\mathit{lookup} \cdot \dot{\lambda}) \\
&\simeq \mathit{find\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda} \cdot (\mathit{lookup} \cdot \dot{\lambda})
\end{aligned}$$

□

We now tackle the main correctness proof for ∇_k.

Proposition 7.12. Suppose $\Phi \in \mathsf{C}_k$, $\nu \Vdash_k^\delta \Phi$, $\dot{\nu} \in \mathsf{Tot}(A_1)$ represents ν, $G \in \mathsf{C}_{k-1}$, and $\dot{G} \in \mathsf{Tot}(A_{k-1})$ represents G. Then $\nabla_k \cdot \dot{\nu} \cdot G = \widehat{\Phi(G)}$.

Proof. Let $\dot{\lambda} = \mathit{satis\text{-}rep} \cdot \dot{\nu}$; then by Proposition 7.9, $\dot{\lambda}$ represents a satisfactory representation λ of Φ. For any j, we may define $\dot{\lambda}^j = \mathit{shift} \cdot \widehat{j} \cdot \dot{\lambda}$, so that $\dot{\lambda}^j$ represents λ^j. Since the element $\mathit{shift} \cdot \widehat{j} \cdot \dot{\lambda}$ is always normalized, for any j, j' we have $\mathit{shift} \cdot \widehat{j} \cdot \dot{\lambda}^{j'} = \dot{\lambda}^{j'+j}$.

By condition (4) of Definition 7.8, there is a position index j_0 such that $\lambda(j_0)$ has the form $2(c \mapsto \breve{q}) + 1$, where

$$c = \langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1} \rangle$$

and each (b_i, p_i) is satisfactory for $\dot{G}$ relative to λ^{j_0+1}. Now by condition (2) of Proposition 7.6, for each i we have $G(h) = p_i$ for all $h \in O_{b_i}$, so $G \in O_c$. So by condition (3) of Definition 7.8 we have $q = \Phi(G)$. Moreover, by condition (1) of Proposition 7.6, we have $\mathit{satisfies\text{-}code} \cdot \dot{G} \cdot \dot{\lambda}^{j_0+1} \cdot \widehat{b_i} \cdot \widehat{p_i} = \mathit{tt}$ for each i, and it follows easily that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j_0+1} \cdot \widehat{\lambda(j_0)} \in \mathcal{T}$.

Set $x_j = \lambda(j)$ for each j. We will show that the following hold for all $j \leq j_0$:

(1) $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \in \mathcal{T} \cup \mathcal{F}$.

(2) If $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \in \mathcal{T}$, then $q_j = \mathrm{result}(x_j) = \Phi(G)$.

We argue by reversed induction on j. For the case $j = j_0$, both claims are established above. So suppose $j < j_0$, and assume both claims hold for all j' where $j < j' \leq j_0$. If x_j is even (either 0 or a token $2(c_j \mapsto \breve{q}_j) + 2$), then $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} = \mathit{ff}$ so that both claims are trivial. So suppose $x_j = 2(c_j \mapsto \breve{q}_j) + 1$.

Using the induction hypothesis, for any $m \leq j_0 - j - 1$ we have

$$\mathit{settles} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{m} \;=\; \mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{m+j+2} \cdot \widehat{x}_{m+j+1} \;\in\; \mathcal{T} \cup \mathcal{F}$$

and if $m = j_0 - j - 1$ then $\mathit{settles} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{m} \in \mathcal{T}$. Moreover, since $c_j \in \mathrm{dom}\ \zeta_{k-1}$, using Proposition 7.10(i) we have $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x_j} \cdot \widehat{m} \in \mathcal{T} \cup \mathcal{F}$ for every $m \in \mathbb{N}$. It follows that $\mathit{pre\text{-}critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j}$ successfully evaluates to some numeral $\widehat{m_0}$ where $m_0 \leq j_0 - j - 1$. Set $m_0' = m_0 + j + 1$.

We next claim that $\mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \cdot \widehat{m_0}$ successfully evaluates to a numeral.

Clearly, $\mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \cdot \widehat{m_0} = \widehat{m_0}$ unless we are in the case where $\mathit{settles} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{m_0} \in \mathcal{T}$ but $q_j \neq q_{m_0'} = \mathrm{result}(x_{m_0'})$. In this latter case, we have $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{m_0'+1} \cdot \widehat{x_{m_0'}} \in \mathcal{T}$, so by claim (2) of the induction hypothesis we have $q_{m_0'} = \Phi(G)$, whence $q_j \neq \Phi(G)$. But by condition (3) of Definition 7.8, $c_j \mapsto \breve{q_j}$ is consistent with Φ, so G cannot be consistent with c_j. So by Proposition 7.10(ii), there must be some smallest m_1 such that $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x_j} \cdot \widehat{m_1} \in \mathcal{T}$, and then $\mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \cdot \widehat{m_0} = \widehat{m_1}$.

By condition (2) of Definition 7.8, we have that $\mathit{lookup} \cdot \dot{\lambda}^{j+1} = \widehat{x_j}$. Using Lemma 7.11, we now have

$$\begin{aligned} \mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} &\succeq \mathit{find\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \\ &\succeq \mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \cdot \\ &\qquad (\mathit{pre\text{-}critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j}) \\ &= \mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \cdot \widehat{m_0} \end{aligned}$$

so $\mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda}^{j+1}$ evaluates to a numeral, $\widehat{m}$ say. By Proposition 7.5(ii), it follows that $\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \dot{\mu}^{b_i}$ is a total element of A_{k-2} for each code b_i appearing in c_j, and hence that each $\dot{G} \cdot (\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \dot{\mu}^{b_i})$ is a numeral, so $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \in \mathcal{T} \cup \mathcal{F}$. This establishes claim (1) for j.

For claim (2), if $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \in \mathcal{T}$ then we must have $\dot{G} \cdot (\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \dot{\mu}^{\dot{b}_i}) = \widehat{p_i}$ for each $t_i = b_i \mapsto \breve{p}_i$ appearing in c_j. But $\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \dot{\mu}^{\dot{b}_i} = \mathit{std\text{-}total} \cdot \widehat{b_i} \cdot \widehat{m}$, so $\mathit{std\text{-}satisfies} \cdot \dot{G} \cdot \widehat{m} \cdot \widehat{t_i} = \mathit{tt}$ for each i. Hence

$$\mathit{refutes} \cdot \dot{G} \cdot \widehat{x_j} \cdot \widehat{m} = \mathit{ff}$$

So by inspection of the definition of `test-pre-critical`, we are in the case where $\mathit{settles} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \in \mathcal{T}$ and $q_{m+j+1} = q_j$. By the definition of `settles`, this means that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{m+j+2} \cdot \widehat{x}_{m+j+1} \in \mathcal{T}$, and by claim (2) of the induction hypothesis, this implies that $q_{m+j+1} = \Phi(G)$. Hence $q_j = \Phi(G)$ as required.

To complete the proof, let j_1 be the least j such that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j} \in \mathcal{T}$ (note that $j_1 \leq j_0$). Then by claim (1) above, we have that $\mathit{sat\text{-}all\text{-}pos} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{j} \in \mathcal{F}$ for all $j < j_1$; hence $\mathit{search} \cdot \dot{\lambda} \cdot \dot{G} = \widehat{j_1}$. But now by claim (2) above we have that $\nabla_k \cdot \dot{\nu} \cdot \dot{G} = \mathit{result} \cdot \widehat{x_{j_1}} = \widehat{\Phi(G)}$ and we are done. □

The proof of Theorems 5.13 and 5.14 is now complete.

8. Proof for the effective case

We now give the proof of our main theorem for HEO (Theorem 5.12). The proof is in many ways analogous to the proof for the continuous case, although some additional work is occasioned by the requirement that certain constructions be performed effectively. Moreover, the two proofs diverge completely at the point where *satisfactory representations* are introduced, since the representations in question are entirely different. (It is interesting that neither of the methods we use at this point seems to be applicable in the other case.) It may be that a more unified approach which abstracts out the common content of the two proofs is possible, though we shall leave this undertaking for future work.

Our account of the proof for the effective case will be technically almost self-contained: rather than constantly referring the reader to Section 7 we will err somewhat on the side of repetition, deferring to Section 7 only where the analogy is quite precise for substantial sections of the proof. However, we will be rather more sparing on motivational remarks where these only repeat what we said for the continuous case.

Throughout this section, we take A to be an NR-TPCA equipped with a realization $\gamma : A \longrightarrow\!\triangleright K_1$, with associated elements $\mathbf{a}_{\sigma\tau}$ and $\mathbf{d}$ as in Definition 5.1, such that conditions (A) and (B) of Theorem 5.12 are satisfied. To establish Theorem 5.12 we wish to prove that $\gamma_*(\mathbf{EC}(A)) \cong \mathbf{HEO}$ in $\mathbf{Asm}(K_1)$.

As in the continuous case, we prove this by means of a lemma asserting the existence of Normann programs for all pure types. In order to formulate the lemma, we make use of the model $\mathsf{P}^{\mathit{eff}}$. Recall from Section 5.3 the near-realization $\Vdash_{\mathrm{B}}^{\mathsf{P}^{\mathit{eff}}}$ of $\mathsf{P}^{\mathit{eff}}$ over K_2^{eff} given by basic enumerations. As observed in Section 5.3, this induces a genuine realization $\Vdash_{\mathrm{B}}^{\mathsf{P}^{\mathit{eff}}} \circ\, \epsilon^{\mathsf{P}^{\mathit{eff}}} : \mathsf{EC}(\mathsf{P}^{\mathit{eff}}) \longrightarrow\!\triangleright K_2^{\mathit{eff}}$. We write δ for the corresponding realization $\mathsf{HEO} \longrightarrow\!\triangleright K_2^{\mathit{eff}}$; then by Theorem 5.10(iii) and the remarks at the end of Section 5.3, the realization $\vdash^{K_2\,\mathit{eff}} \circ\, \delta$ is isomorphic to $\vdash^{\mathsf{HEO}}$.

If $k \geq 1$, and $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{HEO}$ are indeed isomorphic at least up to level $k-1$, then precisely as in the continuous case, for any $l \leq k$ we may define what it means for an element $\dot{x} \in A_l$ to *represent* some $x \in \mathsf{HEO}_l$. Recall that if $\dot{x}$ represents x, then $\dot{x}$ is total in A_l. Likewise, in view of the isomorphism $\mathsf{EC}(\mathsf{P}^{\mathit{eff}}) \cong \mathsf{HEO}$, we have the notion of an element $\mathfrak{x} \in \mathsf{P}_l^{\mathit{eff}}$ *representing* some $x \in \mathsf{HEO}_l$. As in Section 7, if some symbol denotes an element of HEO, we will use its dotted counterpart for an element of A that represents it, and its Gothic counterpart for an element of P that represents it.

The statement of our main lemma is almost identical to that of Lemma 7.1, except that here we work with the language NRComb rather than NRComb$^+$:

Lemma 8.1 (Main Lemma). Suppose $k \geq 1$, and suppose $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{HEO}$ are isomorphic up to level $k-1$. Then there is an NRComb-definable constant $\mathtt{nabla}_k : \overline{1} \to \overline{k}$ denoting an element $\nabla_k \in A_{\overline{1}\to\overline{k}}$ with the following property: whenever $\Phi \in \mathsf{HEO}_k$, $\nu \Vdash_k^{\delta} \Phi$ and $\dot{\nu} \in \mathsf{Tot}(A_1)$ represents ν, the element $\nabla_k \cdot \dot{\nu} \in A_k$ represents Φ within A.

As before, an element ∇_k satisfying the above condition will be called a *Normann operator* for type $\overline{k}$ in A, and the program of NRComb that defines it will be called a *Normann program*.

The proof of the following is now analogous to that of Theorem 7.2.

Theorem 8.2. Assuming Lemma 8.1, we have $\gamma_*(\mathbf{EC}(A)) \cong \mathbf{HEO}$.

Proof. By Theorem 6.6, it suffices to show that $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{HEO}$ are isomorphic at the pure types. We construct suitable isomorphisms $\imath_k$ by induction on k. As in Theorem 7.2, the isomorphism $\imath_0 : \gamma_*(\mathbf{EC}(A))_0 \cong \mathbf{HEO}_0$ is given by the fact that $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{HEO}$ are both internal type structures.

For the induction step, suppose $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{HEO}$ are isomorphic up to level $k-1$ via isomorphisms $\imath_l : \gamma_*(\mathbf{EC}(A))_l \to \mathbf{HEO}_l$, and let $\jmath_l = \imath_l^{-1}$ for each l. We will construct morphisms $\imath_k : \gamma_*(\mathbf{EC}(A))_k \to \mathbf{HEO}_k$ and $\jmath_k : \mathbf{HEO}_k \to \gamma_*(\mathbf{EC}(A))_k$ that commute

with the application morphisms. Since both $\mathsf{EC}(A)$ and HEO are extensional, this will imply that $\imath_k, \jmath_k$ are mutually inverse isomorphisms.

For $\imath_k$, the required morphism arises as the exponential transpose of

$$\imath_0 \circ app_{k-1} \circ (id \times \jmath_{k-1}) \ : \ \gamma_*(\mathbf{EC}(A))_k \times \mathbf{HEO}_{k-1} \longrightarrow \mathbf{HEO}_0$$

and it is clear by construction that this morphism commutes with application. It remains to construct $\jmath_k$. First, by Theorem 5.10(iii), from any e realizing Φ in $\mathbf{HEO}_k$ we may effectively compute a realizer $e' \vdash_k^\delta \Phi$ — that is, a Kleene index for a basic enumeration ν of some total $\mathfrak{Phi} \in \mathsf{P}_k^{\mathit{eff}}$ representing Φ. From e' we may then effectively compute an e'' such that $e'' \vdash_1^\gamma \dot{\nu}$ for some $\dot{\nu} \in A_1$ representing ν. Now let $\widehat{\Phi} = \nabla_k \cdot \dot{\nu} \in A_k$; then using the element $\mathbf{a}_{1k}$ and a realizer for ∇_k, from e'' we may effectively compute a realizer $e''' \vdash_k^\gamma \widehat{\Phi}$. But by Lemma 8.1, $\widehat{\Phi}$ represents Φ in $\mathbf{EC}(A)_k$, and so e''' is a realizer for Φ in $\gamma_*(\mathbf{EC}(A)_k)$. Putting all this together, from a realizer for $\Phi \in \mathbf{HEO}_k$ we have effectively computed a realizer for $\Phi \in \gamma_*(\mathbf{EC}(A)_k)$, so we have a morphism $\jmath_k : \mathbf{HEO}_k \to \gamma_*(\mathbf{EC}(A)_k)$ commuting with application as required. □

The proof of Lemma 8.1 is by induction on k. For $k = 1, 2$, the proofs are exactly analogous to those given in Section 7.1 — just replace C by HEO, K_2 by K_2^{eff}, and P by $\mathsf{P}^{\mathit{eff}}$. (Indeed, precisely the same programs $\mathtt{nabla}_1$ and $\mathtt{nabla}_2$ may be used in the effective setting.) The remainder of this section is therefore devoted to proving the lemma for an arbitrary $k \geq 3$, assuming the lemma for $k-2$.

8.1. *A notion of "partial element" in A_{k-2}*

Suppose $k \geq 3$, and suppose $\gamma_*(\mathbf{EC}(A))$ and $\mathbf{HEO}$ are isomorphic up to level $k-1$. Suppose also, as our induction hypothesis, that there is a Normann program $\mathtt{nabla}_{k-2} : \overline{1} \to \overline{k-2}$ with the property stated in Lemma 8.1; we wish to construct a similar Normann program for type $\overline{k}$.

Suppose $\Phi \in \mathsf{HEO}_k$ and $\nu \vdash_k^\delta \Phi$; then $\nu \in K_2^{\mathit{eff}}$ basically enumerates some $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}_k^{\mathit{eff}})$ which represents Φ. The codes enumerated by ν each have the form

$$\langle \langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1} \rangle \mapsto \breve{q} \rangle$$

where each b_i is a type $k-2$ code of the form $\langle a_{i0} \mapsto \breve{n}_{i0}, \ldots, a_{i(s_i-1)} \mapsto \breve{n}_{i(s_i-1)} \rangle$. Our ultimate task is to construct an element $\nabla_k \in A_{\overline{1} \to \overline{k}}$ which operates on enumerations ν of this kind. To this end, however, we need to introduce an alternative representation of type $k-2$ and $k-1$ compact elements which may appear eccentric at first sight.

Let $\bullet$ denote Kleene application as in Example 3.1, and let $\mathtt{bullet} : \overline{0} \to \overline{0} \to \overline{0}$ be a program that uniformly defines the partial computable function $(m, n) \mapsto m \bullet n$. Whenever $u \bullet 0$ is defined, let $\mathit{time}(u)$ be the number of steps taken to compute $u \bullet 0$ on some universal Turing machine fixed in advance. We adopt the convention that $\mathit{time}(u) = \infty$ if the computation of $u \bullet 0$ diverges, where $n < \infty$ for all $n \in \mathbb{N}$. We write H for the halting set $\{u \mid \mathit{time}(u) < \infty\}$, and $\overline{H}$ for its complement $\{u \mid \mathit{time}(u) = \infty\}$.

We say a number $z = \langle d, u \rangle$ *pre-signifies* a compact element $\mathfrak{b} \in \mathsf{P}_{k-2}^{\mathit{comp}}$ if

— $\mathit{time}(u) < \infty$,

— for all $i < time(u)$, $d \bullet i$ is defined and is a basic type $k-2$ code,
— $\mathfrak{b} = \bigsqcup_{i<time(u)} \zeta_{k-2}(d \bullet i)$.

We may also say that $\langle d, u\rangle$ pre-signifies a *total* element $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$ if $time(u) = \infty$ and d is a Kleene index for some basic enumeration of $\mathfrak{h}$. (Recall from Section 5.3 that such an enumeration must include *all* the basic compacts below $\mathfrak{h}$.) In general we write $[z]$ for the (compact or total) type $k-2$ element pre-signified by z when this exists.

We then say a number $s = \langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle$ *signifies* an element $\mathfrak{c} \in \mathsf{P}^{\mathit{comp}}_{k-1}$ if for each $i < r$, $\langle d_i, u_i\rangle$ pre-signifies some type $k-2$ compact element $\mathfrak{b}_i$, and $\mathfrak{c} = \bigsqcup_{i<r}(\mathfrak{b}_i \Rightarrow p_i)$. We write $[\![\, s \,]\!]$ for the standard type 2 element signified by s when this exists.¶¶ We will also sometimes say that a number s *basically signifies* a type $k-1$ basic compact $\mathfrak{c}$ if s is of the form $\langle\langle d, u, p\rangle\rangle$ and s signifies $\mathfrak{c}$ in the above sense.

Clearly, from a pre-signifier for some $\mathfrak{b} \in \mathsf{P}^{\mathit{comp}}_{k-2}$ we may effectively compute a ζ_{k-2}-code for $\mathfrak{b}$; let $\mathtt{code} : \overline{0} \to \overline{0}$ uniformly represent a partial computable function that does this. Likewise, from a signifier for $\mathfrak{c} \in \mathsf{P}^{\mathit{comp}}_{k-1}$ we may effectively compute a ζ_{k-1}-code for $\mathfrak{c}$. It is also easy to see that the set of all possible pre-signifiers for an element $\zeta_{k-2}(b)$ is c.e. uniformly in b, and (hence) that the set of all possible signifiers for an element $\zeta_{k-1}(c)$ is c.e. uniformly in c.

The purpose of this "signifying representation" is similar to that of the modified enumerations in the continuous setting. We are going to use signifiers $\langle d, u\rangle$ with $time(u) < \infty$ to help us to define "partial elements" in A_{k-2}, so that signifiers with $time(u) = \infty$ correspond to the limiting total elements. The idea is that we know how $\dot{G}$ behaves on the latter, and the undecidability of the halting problem provides us with a kind of "continuity" which gives us a handle on how $\dot{G}$ behaves on partial elements.

We next show how type $k-2$ pre-signifiers can be interpreted as elements of A_{k-2}. We will write a program `interpret` which, given a pre-signifier $z = \langle d, u\rangle$ for a compact element $\mathfrak{b}$, yields an element $\dot{b} \in A_{k-2}$ which somehow plays the role of $\mathfrak{b}$.

Since we may naturally regard the partial computable function ϕ_d as a "partial" basic enumeration of $\mathfrak{b}$, and we already have an operator ∇_{k-2} which turns basic enumerations into elements of A_{k-2}, it might be thought that $\nabla_{k-2} \cdot (\mathit{bullet} \cdot \widehat{d})$ is a suitable candidate for $\dot{b}$. In fact, what we shall pass to ∇_{k-2} will be an improved function which potentially extends ϕ_d. This extension will be constructed with the help of a program

$$\mathtt{critical\text{-}index} \;:\; \overline{k-1} \to \overline{1} \to \overline{0} \to \overline{0}$$

which we will shall leave unspecified for the time being (it will be defined in Section 8.4 below). This program takes three arguments $\dot{G}, \dot{\lambda}, \widehat{e}$, where $\dot{G}$ is the total element of A_{k-1} in question, and the other parameters represent a function $\lambda : \mathbb{N} \to \mathbb{N}$ and a number $e \in \mathbb{N}$ whose roles will be explained later. The program `critical-index` also has the property that $\mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda}\downarrow$ for all $\dot{G}, \dot{\lambda}$.

We first define a thunked version of `critical-index` as follows.

$$\mathtt{critical\text{-}index\text{-}thunk}\ \mathtt{G}^{k-1}\ \mathtt{lam}^{1}\ \mathtt{e}^{0}\ \mathtt{dummy}^{0} \;\equiv\; \lceil \mathtt{critical\text{-}index\ G\ lam} \rceil\ \mathtt{e}$$

¶¶ Strictly speaking, the meaning of $[z]$ and $[\![\, s \,]\!]$ is dependent on k, but we may regard k as being fixed for the remainder of the proof.

In contrast to the definition in Section 7.2, here the protected subexpression on the right hand side includes the argument `lam`; the reason for this will become apparent in the proof of Lemma 8.12. Note that *critical-index-thunk* $\cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e}\downarrow$ for all $\dot{G}, \dot{\lambda}, \widehat{e}$.

Let $\texttt{halts-within} : \overline{0} \to \overline{0} \to \overline{0}$ be a program that uniformly defines the total computable function

$$(i, u) \;\mapsto\; \begin{cases} \mathit{tt} & \text{if } \mathit{time}(u) \leq i \\ \mathit{ff} & \text{otherwise} \end{cases}$$

As before, we suppose $\texttt{below} : \overline{0}^{(2)} \to \overline{0}$ uniformly represents the function

$$(c, c') \;\mapsto\; \begin{cases} \mathit{tt} & \text{if } c, c' \in \mathrm{dom}\ \zeta_{k-2} \text{ and } \zeta_{k-2}(c) \sqsubseteq \zeta_{k-2}(c'), \\ \mathit{ff} & \text{otherwise} \end{cases}$$

and $\texttt{xi} : \overline{0}^{(2)} \to \overline{0}$ defines the total computable function $(c, i) \mapsto \xi_{k-2,c}(i)$.

By analogy with the definitions in Section 7.2, we now define the programs

$$\begin{aligned}
\texttt{filter b}^0\ \texttt{c}^0 \;&\equiv\; \texttt{if (below b c) c b} \\
\texttt{filter-enum z}^0\ \texttt{thunk}^1\ \texttt{i}^0 \;&\equiv\; \texttt{xi (filter (code z) (thunk 0)) i} \\
\texttt{extend-enum z}^0\ \texttt{thunk}^1\ \texttt{i}^0 \;&\equiv\; \texttt{if}_1\ \texttt{(halts-within i (proj1 z))} \\
&\qquad\quad \texttt{(bullet (proj0 z))} \\
&\qquad\quad \texttt{(filter-enum z thunk)} \\
&\qquad\quad \texttt{i} \\
\texttt{partial-elt z}^0\ \texttt{thunk}^1 \;&\equiv\; \texttt{nabla}_{k-2}\ \texttt{(extend-enum z thunk)}
\end{aligned}$$

The essential properties of `partial-elt` are as follows:

Proposition 8.3. (i) If z pre-signifies some $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing $h \in \mathsf{C}_{k-2}$, then for any *thunk* $\in A_1$, *partial-elt*$\cdot\widehat{z}\cdot$*thunk* is an element of $\mathsf{Tot}(A_{k-2})$ representing h (whether or not *thunk* $\cdot\, \widehat{0}\downarrow$).

(ii) If z pre-signifies some $\mathfrak{b} \in \mathsf{P}_{k-2}^{comp}$ and *thunk* $\cdot\, \widehat{0} = \widehat{c}$, then *partial-elt* $\cdot\, \widehat{z} \cdot$ *thunk* is an element of $\mathsf{Tot}(A_{k-2})$ representing some $h \in U_{\mathfrak{b}}$.

(iii) If z pre-signifies some $\mathfrak{b} \in \mathsf{P}_{k-2}^{comp}$, and $\mathfrak{b}'$ is any compact element above $\mathfrak{b}$ so that $U_{\mathfrak{b}'} \subseteq U_{\mathfrak{b}}$, then there exists $c \in \mathbb{N}$ such that, whenever *thunk* $\cdot\, \widehat{0} = \widehat{c}$, *partial-elt* $\cdot\, \widehat{z} \cdot$ *thunk* represents some $h \in U_{\mathfrak{b}'}$.

Proof. Analogous to the proof of Proposition 7.5. □

The following gives us a way of interpreting a pre-signifier z as an element in A_{k-2}:

$$\begin{aligned}
\texttt{interpret G}^{k-1}\ \texttt{lam}^1\ \texttt{e}^0\ \texttt{z}^0 \;&\equiv\; \texttt{partial-elt z} \\
&\qquad\quad (\lceil\texttt{critical-index-thunk G lam}\rceil\ \texttt{e})
\end{aligned}$$

If z pre-signifies some compact element $\mathfrak{b}$, we may regard the element $\dot{b} = \mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{z}$ as a kind of counterpart to $\mathfrak{b}$ within A_{k-2}.

We may therefore consider a triple (d, u, p) as providing a test which we can apply to a total $\dot{G} \in A_{k-1}$. Informally, if $u \in H$ and $\langle d, u\rangle$ signifies $\mathfrak{b}$, the intention is to test whether "$\dot{G} \cdot \mathfrak{b} = \widehat{p}$". As it stands this is nonsensical, but we can at least perform an

analogous test using $\dot{b} \in A_1$ in place of $\mathfrak{b}$. Again, we carry around $\dot{\lambda}$ and $\widehat{e}$ as additional parameters.

$$\begin{aligned} \texttt{G-value}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{e}^0\ \texttt{z}^0 &\equiv \texttt{G}\ (\lceil\texttt{interpret G lam}\rceil\ \texttt{e z}) \\ \texttt{satisfies}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{e}^0\ \texttt{p}^0\ \texttt{d}^0\ \texttt{u}^0 &\equiv \texttt{eq}\ (\lceil\texttt{G-value G lam e}\rceil \\ &\qquad (\texttt{make-pair d u}))\ \texttt{p} \end{aligned}$$

Clearly, if $z = \langle d, u\rangle$ pre-signifies some total $\mathfrak{h}$ representing $h \in \mathsf{HEO}_{k-2}$, then *interpret* $\cdot$ $\dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{z}$ also represents h, and so *satisfies* $\cdot\, \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u}$ will be *tt* if $G(h) = p$, and *ff* otherwise.

8.2. *The "graph" of a total type $k-1$ element*

Our argument now proceeds roughly as in the continuous case, except that in place of the continuity of K_2 we exploit the undecidability of the halting problem, much as in the proofs of the Myhill-Shepherdson and Kreisel-Lacombe-Shoenfield theorems. If we are in the scenario just mentioned and $G(h) = p$, then since $\overline{H}$ is not semidecidable, it must be possible to replace $u \in \overline{H}$ by some $u' \in H$ such that *satisfies* $\cdot\, \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u'}$ is still *tt*. The signifier $\langle d, u', p\rangle$ then defines a kind of "partial element". for $\dot{G}$, which moreover can be computed from d (uniformly in the other parameters). By doing this for every total index d, we may obtain a set of such partial elements that comprise a complete graph for some element of P_{k-1}, whose action on total elements of $\mathsf{P}^{\mathit{eff}}_{k-2}$ corresponds to that of G. Furthermore, we can ensure that our graph is itself effectively generated, so that we obtain an element $\mathfrak{G} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-1})$ representing G.

Once we have such an element $\mathfrak{G}$, our total functional $\mathfrak{Phi} \in \mathsf{P}^{\mathit{eff}}_k$ will know what to do with it — that is, ν must feature some basic compact $\mathfrak{c} \Rightarrow q$ such that $\mathfrak{c} \sqsubseteq \mathfrak{G}$, and this tells us that $\Phi(G) = q$.

We now give the detailed argument leading to the construction of $\mathfrak{G}$ (parametrically in $\dot{\lambda}$ and e). The next two propositions show how a suitable element $u' \in H$ may be obtained. The first of these propositions embodies the essential content of the KLS theorem for type $k-1$; the proof is worthy of some attention since it foreshadows our main argument in certain respects. The programs introduced in the course of the proof will not be required in the sequel.

Proposition 8.4. Suppose given $\dot{G} \in \mathsf{Tot}(A_{k-1})$ and $d \in \mathbb{N}$ such that ϕ_d basically enumerates some $\mathfrak{h}$ representing $h \in \mathsf{HEO}_{k-2}$. Suppose also that $\dot{h} \in \mathsf{Tot}(A_{k-2})$ represents h, and $\dot{G} \cdot \dot{h} = \widehat{p}$. Then there exists a number $u \in H$ which *signifies a KLS modulus of* $\dot{G}$ *at* d, p in the following sense: if $t = \mathit{time}(u)$ and $\mathfrak{b} = \bigsqcup_{i<t} \zeta_{k-2}(d \bullet i)$, then for all $\dot{h}'$ representing any $h' \in U^{\mathsf{HEO}}_{\mathfrak{b}}$ we have that $\dot{G} \cdot \dot{h}' = \widehat{p}$.

Moreover, a suitable value for u may be effectively computed from d and any realizer $y \vdash^{\gamma}_{k-1} \dot{G}$.

Proof. Recall from Theorem 4.9 our standard enumeration $\mathfrak{x}_0, \mathfrak{x}_1, \ldots$ of a dense subset of $\mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$, and let x_m denote the element of HEO_{k-2} represented by $\mathfrak{x}_m$. Also let G be the

element of HEO_{k-1} represented by $\dot{G}$. We first define a program `easy-critical-index` which, given $\dot{G}$ and d as above and $u \in H$, searches through this dense subset in the hope of finding some $x_m \in U_{\mathfrak{b}}^{\mathsf{HEO}}$ such that $G(x_m) \neq p$. (This can be viewed as a rehearsal for the program `critical-index` which we shall eventually define in Section 8.4.)

$$\begin{aligned}
\texttt{std-dense z}^0\ \texttt{m}^0 &\equiv \texttt{partial-elt z (k m)}\\
\texttt{is-counterex-for G}^{k-1}\ \texttt{z}^0\ \texttt{p}^0\ \texttt{m}^0 &\equiv \texttt{not (eq (G (std-dense z m)) p)}\\
\texttt{easy-critical-index G}^{k-1}\ \texttt{z}^0\ \texttt{p}^0 &\equiv \texttt{min (is-counterex-for G z p)}
\end{aligned}$$

We may then define a program which tests whether in fact $G(x_m) = p$ for the index m obtained in this way. Our definition mimics the above definition of `satisfies`.

$$\begin{aligned}
\texttt{easy-crit-ind-thunk G}^{k-1}\ \texttt{z}^0\ \texttt{p}^0\ \texttt{dummy}^0 &\equiv \texttt{easy-critical-index G z p}\\
\texttt{critical-elt G}^{k-1}\ \texttt{z}^0\ \texttt{p}^0 &\equiv \texttt{partial-elt z}\\
&\qquad \texttt{(easy-crit-ind-thunk G z p)}\\
\texttt{easy-sat G}^{k-1}\ \texttt{z}^0\ \texttt{p}^0 &\equiv \texttt{eq (G (critical-elt G z p)) p}\\
\texttt{easy-satisfies G}^{k-1}\ \texttt{p}^0\ \texttt{d}^0\ \texttt{u}^0 &\equiv \texttt{easy-sat G (make-pair d u) p}
\end{aligned}$$

If $u \in \overline{H}$ then by Proposition 8.3(i) we have that $\mathit{critical\text{-}elt} \cdot \dot{G} \cdot \widehat{\langle d, u\rangle} \cdot \widehat{p}$ represents h, so that $\mathit{easy\text{-}satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u} = \mathit{tt}$. We will show using the undecidability of the halting set that there must also exist some $u \in H$ satisfying this equation.

Let $Q = \mathit{easy\text{-}satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot \widehat{d}$; take $q \vdash_1^\gamma Q$; let $\psi : \mathbb{N} \to \mathbb{N}$ be total computable such that $\psi(u) \vdash_0^\gamma \widehat{u}$ for all u; and let $R = \{r \in \mathbb{N} \mid r \vdash_0^\gamma \mathit{tt}\}$, recalling that $\mathit{tt} = \widehat{0}$. Then for any $u \in \overline{H}$ we have $Q \cdot \widehat{u} = \mathit{tt}$ and so $\mathbf{a}_{00} \bullet q \bullet \psi(u) \in R$. But the set R is c.e. by condition (B) of Theorem 5.12, so by the undecidability of the halting problem, there must exist some $u \in H$ such that $\mathbf{a}_{00} \bullet q \bullet \psi(u) \in R$. Now by condition (A) of Theorem 5.12, we can conclude that $Q \cdot \widehat{u} = \mathit{tt}$ as required. Moreover, since R is c.e., a suitable $u \in H$ may be obtained effectively from d and y by means of a simple search. Note that p may be readily computed from y and d since $\widehat{p} = \dot{G} \cdot (\nabla_{k-2} \cdot (\mathit{bullet} \cdot \widehat{d}))$.

We now show that any $u \in H$ such that $\mathit{easy\text{-}satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u} = \mathit{tt}$ signifies a KLS modulus of $\dot{G}$ at d, p. Let $\mathfrak{b}$ be as in the proposition. We first show that for each $x_m \in U_{\mathfrak{b}}^{\mathsf{HEO}}$ we have $G(x_m) = p$. Otherwise, let m_0 be the smallest number such that $x_{m_0} \in U_{\mathfrak{b}}^{\mathsf{HEO}}$ but $G(x_{m_0}) \neq p$. Since $G(\xi_m)$ is a numeral for every m, the value of m_0 will be found by *easy-critical-index*: that is, $\mathit{easy\text{-}critical\text{-}index} \cdot \dot{g} \cdot \widehat{z} \cdot \widehat{p} = \widehat{m_0}$, where $z = \langle d, u\rangle$. It follows that $\mathit{critical\text{-}elt} \cdot \dot{G} \cdot \widehat{z} \cdot \widehat{p}$ represents x_{m_0}, and hence that $\mathit{easy\text{-}satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u} = \mathit{ff}$, a contradiction.

Now suppose that for some arbitrary $h' \in U_{\mathfrak{b}}^{\mathsf{HEO}}$ and $\dot{h}'$ representing h, we have $\dot{G} \cdot \dot{h}' = \widehat{p'} \neq \widehat{p}$. Then by the above argument with h', p' in place of h, p, we obtain a finite element $\mathfrak{b}'$ with $h' \in U_{\mathfrak{b}'}^{\mathsf{HEO}}$ such that $G(x_m) = p'$ whenever $x_m \in U_{\mathfrak{b}'}^{\mathsf{HEO}}$. But since $h' \in U_{\mathfrak{b}}^{\mathsf{HEO}} \cap U_{\mathfrak{b}'}^{\mathsf{HEO}}$, we may take m such that $x_m \in U_{\mathfrak{b}}^{\mathsf{HEO}} \cap U_{\mathfrak{b}'}^{\mathsf{HEO}}$. But then $p = G(x_m) = p'$, a contradiction. □

The following proposition is the analogue of Proposition 7.6; it combines the above proposition with information about the behaviour of the program `satisfies`.

Proposition 8.5. Suppose given $e \in \mathbb{N}$, $\dot{\lambda} \in \mathsf{Tot}(A_1)$, and $\dot{G}, d, \mathfrak{h}, h, \dot{h}, p$ as in Proposition 8.4. Then there exists $u' \in H$ such that

(1) $\mathit{satisfies} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u'} = tt$ (that is, "$\dot{G}$ thinks $\langle d, u' \rangle$ is a neighbourhood for the continuity of G at h").

(2) u' signifies a KLS modulus of $\dot{G}$ at d, p in the sense of Proposition 8.4 (that is, "$\langle d, u' \rangle$ really is such a neighbourhood as far as total elements of A_{k-2} are concerned").

Moreover, a suitable value for u' may be effectively computed from e, d, any realizer $w \vdash_1^\gamma \lambda$, and any realizer $y \vdash_{k-1}^\gamma \dot{G}$.

Proof. The previous proposition shows how we may effectively obtain a number u satisfying condition (2). Let $t = \mathit{time}(u)$; then clearly any u' with $\mathit{time}(u') \geq t$ also satisfies condition (2). It therefore suffices to find $u' \in H$ satisfying (1), ensuring that $\mathit{time}(u') \geq t$.

The argument is very similar to the one in the preceding proof. For any $u \in \overline{H}$, using Proposition 8.3(i) we have that $\mathit{satisfies} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u} = tt$, so as in the preceding proof one may construct a function $\chi : \mathbb{N} \rightharpoonup \mathbb{N}$, partial computable uniformly in e, d, w, y, such that $\chi(u) \in R$ for all $u \in \overline{H}$. By condition (B) of Theorem 5.12 and the undecidability of the halting problem, there must be some $u' \in H$ such that $\chi(u') \in R$ and moreover $\mathit{time}(u') \geq t$, and by condition (A) we then obtain condition (1) of the proposition. Moreover, a suitable value of u' may be obtained effectively from e, d, w, y by means of a simple search. (For the time being at least, one should think of the computation of u' as happening "outside A", since the numerical realizers w and y would not be visible within A itself.) □

Note that Propositions 8.4 and 8.5 are the only places in the proof of Theorem 5.12 where conditions (A) and (B) are required.

Some terminology and notation associated with the above proposition will be useful. We will say a type $k-1$ basic signifier $\langle\langle d, u', p \rangle\rangle$ is *satisfactory* for $\dot{G}$ (relative to $\dot{\lambda}$ and e) if $\dot{G}, \dot{\lambda}, e, d, u'$ and p satisfy conditions (1) and (2) of Theorem 8.5. Given e, d, w, y as in the proposition, we write $u(e, d, w, y)$ for the corresponding value of u' computed from them as in the proof above. We also set $\mathfrak{b}(e, d, w, y) = \bigsqcup_{i<t} \zeta_{k-2}(d \bullet i)$ where $t = \mathit{time}(u(e, d, w, y))$. If e, w, y are fixed and $\mathfrak{b} = \mathfrak{b}(e, d, w, y)$, we may think of $U_\mathfrak{b}$ as the *neighbourhood arising from* d; by construction, we clearly have that $G(h') = G(h)$ for all $h' \in U_\mathfrak{b}$.

We have thus shown that there is a plentiful supply of type $k-1$ basic compacts $\mathfrak{c}$ that possess a basic signifier $\langle\langle d, u, p \rangle\rangle$ which is satisfactory for $\dot{G}$. However, some significant further work is needed to show that our enumeration ν of $\mathfrak{Phi}$ must feature a code $\langle\langle c_0, \ldots, c_{r-1} \rangle \mapsto \breve{q} \rangle$ such that the elements $\zeta_{k-1}(c_i)$ *all* have satisfactory signifiers for $\dot{G}$. To establish this, we will show how to generate a complete "c.e. graph" for G consisting entirely of satisfactory signifiers.

The idea is that by applying the Proposition 8.5 to enough indices d, we can generate enough satisfactory signifiers $\langle d, u, p \rangle$ to specify the whole of G. Obviously we could achieve this by letting d range over all possible indices for basic enumerations of total elements, but this will not do since we require our satisfactory signifiers to be effectively

enumerated. We therefore work with a c.e. family of enumerations that yield a dense set in HEO_{k-2}, as in Theorem 4.9. However, if this is done in a naive way (*e.g.* using the enumerations $\xi_{k-2,c}$ of Theorem 4.9), there is no guarantee that the corresponding KLS neighbourhoods will cover the whole of HEO_{k-2}. We overcome this problem by a more careful choice of indices d and a further use of the signifying representation.

Let $\zeta = \zeta_{k-2}$, and consider the sets

$$\begin{aligned} I &= \{(d,v) \in \mathbb{N}^2 \mid \forall i < \mathit{time}(v).\ \zeta(d \bullet i)\!\downarrow\ \wedge \\ &\qquad \forall i,j < \mathit{time}(v).\ \zeta(d \bullet i), \zeta(d \bullet j) \text{ are consistent } \wedge \\ &\qquad \text{if } \mathit{time}(v) = \infty \text{ then } \bigsqcup_i \zeta(d \bullet i) \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})\} \\ J &= \{(d,v) \in I \mid \mathit{time}(v) < \infty\} \end{aligned}$$

Clearly J is c.e. but I is not. For any $(d,v) \in I$, define a total function $\eta_{d,v}$ by

$$\eta_{d,v}(i) \;=\; \begin{cases} d \bullet i & \text{if } i < \mathit{time}(v) \\ \xi_{k-2,c}(i) & \text{if } i \geq \mathit{time}(v) \end{cases}$$

where c is a code for $\bigsqcup_{i<\mathit{time}(v)} \zeta(d \bullet i)$ computed in an evident way from d and v, and $\xi_{k-2,c}$ is as in Theorem 4.9. If $\mathit{time}(v) = \infty$ then $\eta_{d,v}(i) = d \bullet i$ for all i, so $\eta_{d,v}$ basically enumerates the same total element as does ϕ_d. If $\mathit{time}(v) < \infty$ and c is as above, then $\eta_{d,v}$ and $\xi_{k-2,c}$ both basically enumerate the same total element $\mathfrak{x}_c$, since $\zeta(d \bullet i) \sqsubseteq \zeta(c)$ for each $i < \mathit{time}(v)$, and $\xi_{k-2,c}$ satisfies condition (3) of Theorem 4.9. So in either case, $\eta_{d,v}$ is a basic enumeration for some total element of $\mathsf{P}^{\mathit{eff}}_{k-2}$. Moreover, using the effectivity part of Theorem 4.9, from any $(d,v) \in I$ we may effectively obtain a Kleene index $\delta(d,v)$ for $\eta_{d,v}$.

Let D be the set of all indices $\delta(d,v)$ where $(d,v) \in J$. Then D is c.e., and the following facts show that D suffices to generate a complete graph for G.

Lemma 8.6. Suppose $e \in \mathbb{N}$, $\dot{\lambda} \in \mathsf{Tot}(A_1)$, $\dot{G} \in \mathsf{Tot}(A_{k-1})$, $w \vdash^{\gamma}_1 \dot{\lambda}$ and $y \vdash^{\gamma}_{k-1} \dot{G}$. Then for any $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$ there is some index $d \in D$ such that $\mathfrak{b}(e,d,w,y) \sqsubseteq \mathfrak{h}$.

Proof. For any $d' \in \mathbb{N}$ indexing a basic enumeration of some $\mathfrak{h}' \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$, let us define

$$N_{d'} \;=\; \{d'' \in \mathbb{N} \mid\ \forall\, i < \mathit{time}(u(e,d',w,y)).\ d'' \bullet i = d' \bullet i\}$$

Let d_0 be a Kleene index for a basic enumeration of $\mathfrak{h}$. Then for any $v \in \overline{H}$, $\delta(d_0,v)$ is also an index for this basic enumeration, and obviously $d_0 \in N_{\delta(d_0,v)}$. Moreover, the set

$$V_{d_0} \;=\; \{v \in \mathbb{N} \mid d_0 \in N_{\delta(d_0,v)}\}$$

is clearly c.e.: to affirm $v \in V_{d_0}$, calculate $d'_0 = \delta(d_0,v)$, compute the corresponding value $u = u(e,d'_0,w,y)$, and check that $d'_0 \bullet i = d_0 \bullet i$ for all $i < \mathit{time}(u)$. So by the undecidability of the halting problem, there must exist some $v_0 \in H \cap V_{d_0}$. But now $(d_0,v_0) \in J$, so let $d = \delta(d_0,v_0)$. Then $d_0 \in N_d$, so setting $t = \mathit{time}(u(e,d,w,y))$ we have $\mathfrak{b}(e,d,w,y) = \bigsqcup_{i<t} \zeta(d_0 \bullet i) \sqsubseteq \bigsqcup_i \zeta(d_0 \bullet i) = \mathfrak{h}$ as required.

In fact, a suitable index d can be effectively computed from e, w, y and d_0, but we shall not require this information. □

Proposition 8.7. Given $e \in \mathbb{N}$, $\dot{\lambda} \in A_1$ and $\dot{G} \in \mathsf{Tot}(A_{k-1})$ representing $G \in \mathsf{HEO}_{k-1}$, there is a sequence $c_0, c_1, \ldots$ of basic type $k-1$ codes such that

(1) each $\zeta_{k-1}(c_j)$ has a basic signifier $\langle\langle d_j, u_j, p_j\rangle\rangle$ that is satisfactory for $\dot{G}$ relative to $\dot{\lambda}$ and e;
(2) the elements $\zeta_{k-1}(c_j)$ are pairwise consistent;
(3) the element $\mathfrak{G} = \bigsqcup_j \zeta_{k-1}(c_j)$ is total in $\mathsf{P}^{\mathit{eff}}_{k-1}$ and represents G.

Moreover, the codes c_j, and suitable associated signifiers $\langle d_j, u_j, p_j\rangle$, are computable from j uniformly in $e, w \vdash^{\gamma}_1 \dot{\lambda}$ and $y \vdash^{\gamma}_{k-1} \dot{G}$.

Proof. Suppose e, w, y are as above. Let $d_0, d_1, \ldots$ be a computable enumeration of the set D, and for each j, set $u_j = u(e, d_j, w, y)$ and define p_j by $\widehat{p_j} = \dot{G} \cdot (\nabla_{k-2} \cdot (\mathit{bullet} \cdot \widehat{d_j}))$. Then $s_j = \langle\langle d_j, u_j, p_j\rangle\rangle$ is a satisfactory type $k-1$ signifier for $\dot{G}$ relative to $\dot{\lambda}$ and e. Let c_j be a basic code for the element $[\![s_j]\!]$, computed in some standard way from d_j, u_j, p_j. The property (1) above and the uniform computability requirement are then obvious.

To show that the elements $\zeta_{k-1}(c_j)$ are pairwise consistent, note that each c_j has the form $b_j \mapsto \breve{p}_j$. Suppose $\zeta_{k-2}(b_j), \zeta_{k-2}(b_{j'})$ are consistent, otherwise $\zeta_{k-1}(c_j), \zeta_{k-1}(c_{j'})$ are trivially consistent. Let $\mathfrak{b} = \zeta_{k-2}(b_j) \sqcup \zeta_{k-2}(b_{j'})$, and take $\mathfrak{x} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$ with $\mathfrak{b} \sqsubseteq \mathfrak{x}$ as in Theorem 4.9. Suppose $\mathfrak{x}$ represents $x \in \mathsf{HEO}_{k-2}$; then $x \in O_{k-2,b_j}$ by Proposition 4.10(ii). Hence $G(x) = p_j$ since $\langle\langle d_j, u_j, p_j\rangle\rangle$ is satisfactory for $\dot{G}$ relative to e and $\dot{\lambda}$. But similarly $G(x) = p_{j'}$, so $p_j = p_{j'}$ and $\zeta_{k-1}(c_j), \zeta_{k-1}(c_{j'})$ are consistent.

We may therefore define $\mathfrak{G} = \bigsqcup_j \zeta_{k-1}(c_j)$. To see that $\mathfrak{G}$ represents G, suppose $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$ represents $h \in \mathsf{HEO}_{k-2}$. By Lemma 8.6 there is some j such that $\mathfrak{b}_{d_j} \sqsubseteq \mathfrak{h}$, whence $\zeta_{k-1}(c_j)(\mathfrak{h}) = p_j = G(h)$. But $\zeta_{k-1}(c_j) \sqsubseteq \mathfrak{G}$, so $\mathfrak{G}(\mathfrak{h}) = G(h)$. □

The following corollary is what the generated graph of G is needed for, and is all that needs to be carried forward from this subsection to the remainder of the proof. Recall that $\Phi \in \mathsf{HEO}_k$ and $\nu \in K_2^{\mathit{eff}}$ is a basic enumeration of some $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_k)$ representing Φ. A number $j \in \mathbb{N}$ playing the role of an argument to ν will be referred to as a *position index* in ν.

Corollary 8.8. For any $e \in \mathbb{N}$, $\dot{\lambda} \in \mathsf{Tot}(A_1)$ and $\dot{G} \in \mathsf{Tot}(A_{k-1})$ representing $G \in \mathsf{HEO}_{k-1}$, there exist a position index j and a type $k-1$ signifier

$$s \;=\; \langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle$$

such that

(1) $\nu(j)$ is a basic code $\langle c \mapsto \breve{q}\rangle$, where $\zeta_{k-1}(c) = [\![s]\!]$ and $q = \Phi(G)$;
(2) for each $i < r$, $\langle\langle d_i, u_i, p_i\rangle\rangle$ is satisfactory for $\dot{G}$ relative to $\dot{\lambda}$ and e.

Moreover, suitable values of j and s, and hence the value $q = \Phi(G)$, are computable from $\nu \in \mathbb{N}^{\mathbb{N}}$, $e \in \mathbb{N}$, $w \vdash^{\gamma}_1 \dot{\lambda}$ and $y \vdash^{\gamma}_{k-1} \dot{G}$ by means of a type 2 partial computable functional (see Section 2.2).

Proof. Fix $e, \dot{\lambda}, \dot{G}$, $w \vdash^{\gamma}_1 \dot{\lambda}$ and $y \vdash^{\gamma}_{k-1} \dot{G}$, and let $\mathfrak{G}$ and $c_0, c_1, \ldots$ be generated as in Proposition 8.7. Since $\mathfrak{Phi}(\mathfrak{G}) = \Phi(G)$ and $\mathfrak{G} = \bigsqcup_j \zeta_{k-1}(c_j)$, there must be some finite r such that if $\mathfrak{c} = \bigsqcup_{i<r} \zeta_{k-1}(c_i)$ then $\mathfrak{Phi}(\mathfrak{c}) = \Phi(G)$. In other words, the step function $\mathfrak{c} \Rightarrow \Phi(G)$ is a basic compact below $\mathfrak{Phi}$, and hence arises as $\zeta_k(\nu(j))$ for some position

index j. Here $\nu(j)$ has the form $\langle c \mapsto \breve{q}\rangle$, where $q = \Phi(G)$ and c is a code for $\mathfrak{c}$. Now each c_i comes together with a satisfactory signifier $\langle\langle d_i, u_i, p_i\rangle\rangle$ as in Proposition 8.7. Let $s = \langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle$; then conditions (1) and (2) are clearly satisfied.

To see how j and s may be effectively computed from ν, e, w, y, recall from Section 8.1 that if c is a type 2 code, the set of all signifiers for $\zeta_{k-1}(c)$ is c.e. uniformly in c. By interleaving enumerations of signifiers for each $\zeta_{k-1}(c_j)$ (where $\nu(j) = \langle c_j \mapsto \breve{q}_j\rangle$), we may enumerate, computably in ν, the set of all pairs (j, s) such that s signifies $\zeta_{k-1}(c_j)$. Moreover, for any such pair, we can effectively test whether s is of the form $\langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle$ where the d_i, u_i, p_i are as in Proposition 8.7, since the sequence of such signifiers is computable uniformly in e, y. But by the above, there exist some j, s for which this is the case, so we can effectively find such a j and s by means of a simple search. Clearly this j and s will satisfy all the required conditions, and we can then effectively obtain $\Phi(G)$ by extracting q_j from $\nu(j)$. □

8.3. *Satisfactory representations for type k objects*

We now turn our attention to the problem of computing $\Phi(G)$ within A itself, given $\dot{\nu}$ and $\dot{G}$ as above. To a crude approximation, what we would like to do is to generate from ν an enumeration of all possible "type k signifiers" $\langle s, q\rangle$ for all codes $\langle c \mapsto q\rangle$ appearing in ν (where $\zeta_{k-1}(c) = [\![s]\!]$), and then test these in turn looking for a signifier

$$\langle\langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle, q\rangle$$

such that each $\langle\langle d_i, u_i, p_i\rangle\rangle$ is satisfactory for $\dot{G}$. When such a signifier is found, we return q. The idea is that Corollary 8.8 should guarantee that the desired signifier exists.

One problem with this idea has to do with the parameter e. The purpose of e is to keep track of where we have got to in our search through the signifiers $\langle s, q\rangle$, and it is essential to the way the Normann programs work that this parameter be passed down to *interpret* and hence to *critical-index* (see Section 8.4 below). There is therefore the *a priori* possibility that the "partial element" $\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot e \cdot \langle d_i, u_i\rangle \in A_1$, and hence that the behaviour of $\dot{G}$ when applied to this partial element, might depend on e. This is why we needed to build the dependency on e into the results of Section 8.2.

The situation is therefore somewhat slippery: for any *fixed* e, the above procedure will eventually throw up a signifier $\langle s, q\rangle$ which is satisfactory for $\dot{G}$ relative to e, but there is no obvious guarantee that there is any e such that a satisfactory signifier for $\dot{G}$ relative to e will occur at precisely the position in the enumeration specified by e.

We can overcome this problem by arranging for the enumeration of the type k signifiers $\langle s, q\rangle$ to be conducted in a special way. This motivates the following definition:

Definition 8.9. A *satisfactory representation* of Φ is a total computable function $\lambda : \mathbb{N} \to \mathbb{N}$, where $\lambda(j) = \langle e_j, s_j, q_j\rangle$ for each j, such that

(1) any number $e \in \mathbb{N}$ appears as e_j for at most one j;
(2) each s_j is a signifier $\langle\langle d_{j0}, u_{j0}, p_{j0}\rangle, \ldots, \langle d_{j(r_j-1)}, u_{j(r_j-1)}, p_{j(r_j-1)}\rangle\rangle$ for some compact element $\mathfrak{c}_j \in \mathsf{P}_{k-1}$;

(3) for each j, the basic compact $\mathfrak{c}_j \Rightarrow q_j$ is consistent with Φ, *i.e.* whenever $G \in U_{\mathfrak{c}_j}$ we have $\Phi(G) = q_j$;
(4) for any $\dot{\lambda}' \in \mathsf{Tot}(A_1)$ and $\dot{G} \in \mathsf{Tot}(A_{k-1})$ representing any $G \in \mathsf{HEO}_{k-1}$, there exists a position index j in λ such that for each $i < r_j$, $\langle d_{ji}, u_{ji}, p_{ji}\rangle$ is satisfactory for $\dot{G}$ relative to $\dot{\lambda}'$ and e_j.

If λ is a satisfactory representation, we will refer to the numbers $\lambda(j) = \langle e_j, s_j, q_j\rangle$ as *tokens* appearing in λ.

The purpose of the e_j is to serve as markers allowing us to locate the current position in λ. Since each e occurs at most once as a marker, given e we may recover our position in λ simply by searching for j such that $e_j = e$. The reader may wonder why we do not simply use the position index j to keep track of our position. The answer is that our more elaborate scheme gives us some extra flexibility which enables us to give an effective construction of satisfactory representations:

Proposition 8.10. Suppose given $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}_k^{\mathit{eff}})$ representing $\Phi \in \mathsf{HEO}_k$, and suppose $\nu \in K_2^{\mathit{eff}}$ is a basic enumeration of $\mathfrak{Phi}$. Then a satisfactory representation λ^ν of Φ exists and is computable from ν via a type 2 partial computable functional.

Proof. Fix Φ and ν. The trick is to use as our place markers e all numbers of the form $\langle y, w\rangle$, where y is (potentially) a γ-realizer for some $\dot{G} \in A_{k-1}$, and w is (potentially) a γ-realizer for some $\dot{\lambda}' \in A_1$. If $e_j = \langle y, w\rangle$, the corresponding s_j and q_j will be chosen specifically to ensure that condition (4) of Definition 8.9 holds at least for the corresponding elements $\dot{G}, \dot{\lambda}'$. By doing this for all possible $\langle y, w\rangle$, all elements $\dot{G} \in \mathsf{Tot}(A_{k-1})$ and $\dot{\lambda}' \in \mathsf{Tot}(A_1)$ will be catered for.

Formally, let $\theta : \mathbb{N}^{\mathbb{N}} \times \mathbb{N}^3 \rightharpoonup \mathbb{N}^3$ be the type 2 partial computable function implicit in the proof of Corollary 8.8, so that whenever $e \in \mathbb{N}$, $w \vdash_1^\gamma \dot{\lambda}'$ and $y \vdash_{k-1}^\gamma \dot{G}$ for some $\dot{\lambda}' \in \mathsf{Tot}(A_1)$ and $\dot{G}$ representing G, we have that $\theta(\nu, e, y, w) = (j, s, q)$, where j, s are as in Corollary 8.8 and $q = \Phi(G)$. Now consider the set

$$M^\nu \;=\; \{\langle e, s, q\rangle \mid \exists y, w, j.\;\; e = \langle y, w\rangle \wedge \theta(\nu, e, y, w) = (j, s, q)\}$$

(It will not matter that this set may feature place markers $\langle y, w\rangle$ where y, w do not realize suitable elements $\dot{G}, \dot{\lambda}'$.) By elementary results of computability theory relative to an oracle for ν, we may show that M^ν is c.e. uniformly in ν. Let λ^ν be a total function enumerating M^ν; clearly λ^ν may be taken to be computable uniformly in ν. Moreover, it is clear by construction that λ^ν satisfies conditions (1) and (2) of Definition 8.9. By inspection of the proof of Corollary 8.8, condition (3) can be seen to hold even when the jth element in λ^ν is generated using some e not of the form $\langle y, w\rangle$ where y, w realize some $\dot{G}, \dot{\lambda}'$ respectively. For condition (4), note that every $\dot{G} \in \mathsf{Tot}(A_{k-1})$ has at least one γ-realizer y and every $\dot{\lambda}' \in \mathsf{Tot}(A_1)$ at least one γ-realizer w, and $\theta(\nu, \langle y, w\rangle, y, w)$ is certainly defined for such a y and w, so a suitable token $\langle\langle y, w\rangle, s, q\rangle$ will eventually turn up in λ^ν. □

By Proposition 2.12, one may construct an NRComb program `satis-rep` : $\overline{1} \to \overline{1}$ that represents the type 2 partial computable function $(\nu, j) \mapsto \lambda^\nu(j)$, so that whenever $\dot{\nu} \in A_1$

represents a basic enumeration ν of some $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}_k^{eff})$, the element $\mathit{satis\text{-}rep} \cdot \dot{\nu} \in A_1$ represents λ^ν.

8.4. *The Normann search algorithm*

Having shown how to compute the required satisfactory representations of Φ, we may now start to assemble the main program for computing $\Phi(G)$ itself. From here on, our proof closely shadows the argument in (Normann 2000). We refer to Section 7.5 for further motivational remarks.

Given $\dot{\nu}$ and $\dot{G}$, our main program will search through the tokens enumerated by λ^ν, looking for a type 2 signifier s that is "satisfied" by $\dot{G}$. The following program `search` returns the position index of the first such signifier. The parameter `x` here stands for a token $\langle e, s, q\rangle$ appearing in λ^ν.

$$\begin{aligned}
\texttt{sat}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{x}^0\ \texttt{t}^0 &\equiv \lceil\texttt{satisfies G lam}\rceil\ (\texttt{proj0 x}) \\
&\qquad (\texttt{proj2 t})\ (\texttt{proj0 t})\ (\texttt{proj1 t}) \\
\texttt{sat-all}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{x}^0 &\equiv \texttt{all-in-list}\ (\lceil\texttt{sat G lam}\rceil\ \texttt{x})\ (\texttt{proj1 x}) \\
\texttt{sat-all-pos}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{j}^0 &\equiv \texttt{sat-all G lam}\ (\texttt{lam j}) \\
\texttt{search}\ \texttt{lam}^1\ \texttt{G}^{k-1} &\equiv \texttt{min}\ (\texttt{sat-all-pos G lam})
\end{aligned}$$

The Normann program itself may then be defined as follows:

$$\texttt{nabla}_k\ \texttt{nu}^1\ \texttt{G}^{k-1} \equiv \texttt{proj2}\ (\texttt{search}\ (\texttt{satis-rep nu})\ \texttt{G})$$

To complete the definition of $\texttt{nabla}_k$, it remains to supply the program

$$\texttt{critical-index} : \overline{k-1} \to \overline{1} \to \overline{0} \to \overline{0}$$

which was left unspecified in Section 7.2. We will define this program in such a way that in all relevant instances the following properties will hold:

(1) The element $\dot{h} = \mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{\langle d, u\rangle}$ is a total element of A_1. This will ensure that $\dot{G} \cdot \dot{h}$ is a genuine numeral, and hence that $\mathit{sat} \cdot \dot{G} \cdot \dot{\lambda} \cdot x \cdot t$ is a genuine truth value.

(2) If $\lambda(j) = \langle e, s, q\rangle$ where $s = \langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle$, and $\dot{G} \cdot (\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{\langle d_i, u_i\rangle}) = \widehat{p_i}$ for each $i < r$, then $\Phi(G) = q$. Together with (1), this will ensure that whenever $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot x$ comes out as true, the corresponding value q returned is indeed the correct result.

The algorithm behind `critical-index` is essentially the same as in the continuous case, so the informal explanation given in Section 7.5 applies *mutatis mutandis*. Naturally, the details of the implementation need to be adapted to cater for the use of the signifying representation rather than the modified enumerations used in Section 7, and also to accommodate the very different notion of "satisfactory representation" used here. Perhaps the most significant difference is that in the continuous case the satisfactory representation λ was shifted at each stage so that only the portion not yet inspected was passed as a parameter, whereas here we keep λ fixed throughout, and instead pass around the markers e in order to keep track of the current position in λ. (In the terminology of Section 5.4, this enables us to dispense with condition (C) in the effective case.)

We now embark on the formal definition of `critical-index`. We take care not to use anything depending on `satis-rep`, since `critical-index` itself enters into the definition of this program.

The machinery required for the first search is very similar that in Section 7.5, the only variations being matters of coding arising from the different form of tokens:

$$\begin{aligned}
\texttt{std-total d}^0\ \texttt{u}^0\ \texttt{m}^0 &\equiv \texttt{partial-elt (make-pair d u) (k m)}\\
\texttt{std-satisfies G}^{k-1}\ \texttt{m}^0\ \texttt{t}^0 &\equiv \texttt{eq (G (std-total (proj0 t) (proj1 t) m))}\\
&\qquad\qquad \texttt{(proj2 t)}\\
\texttt{refutes G}^{k-1}\ \texttt{x}^0\ \texttt{m}^0 &\equiv \texttt{not (all-in-list (std-satisfies G m)}\\
&\qquad\qquad\qquad \texttt{(proj1 x))}
\end{aligned}$$

Proposition 8.11. Suppose $x = \langle e, s, q\rangle$ where s is a type $k-1$ signifier, and $\dot{G} \in \mathsf{Tot}(A_{k-1})$ represents $G \in \mathsf{HEO}_{k-1}$. Then

(i) for any $m \in \mathbb{N}$ we have $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x} \cdot \widehat{m} \in \mathcal{T} \cup \mathcal{F}$;

(ii) if $G \notin U_{[\![s]\!]}$, there exists m such that $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x} \cdot \widehat{m} \in \mathcal{T}$.

Proof. Exactly analogous to the proof of Proposition 7.10. □

We now build up the machinery required for the "second search". Given a marker e arising in a satisfactory representation λ, the following programs allow us to compute the position index at which e occurs in λ, and hence the token $\langle e, s, q\rangle$ in which it appears.

$$\begin{aligned}
\texttt{eq-proj0 lam}^1\ \texttt{e}^0\ \texttt{j}^0 &\equiv \texttt{eq e (proj0 (lam j))}\\
\texttt{position lam}^1\ \texttt{e}^0 &\equiv \texttt{min (eq-proj0 lam e)}\\
\texttt{lookup lam}^1\ \texttt{e}^0 &\equiv \texttt{lam (position lam e)}
\end{aligned}$$

Suppose λ is a satisfactory representation of some $\Phi \in \mathsf{HEO}_k$, $\dot{\lambda}$ represents λ, and $\lambda(j) = x = \langle e, s, q\rangle$. Then by condition (1) of Definition 8.9 we have $\mathit{position} \cdot \dot{\lambda} \cdot \widehat{e} = \widehat{j}$, and $\mathit{lookup} \cdot \dot{\lambda} \cdot \widehat{e} = \widehat{x}$.

The remainder of the definition involves a crucial use of recursion. As in the continuous case, we first present the definition in a circular fashion, and then explain how the circularity may be removed.

The following program (which invokes `sat-all`) is used for the "second search"; it tests whether m is a position index for some later token $\langle e, s, q\rangle$ in λ that can be used to settle the value of $\Phi(G)$. Here j is the "current position index". It is important here that thunking is used so that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e}$ is not computed when $m \leq j$.

$$\begin{aligned}
\texttt{settles G}^{k-1}\ \texttt{lam}^1\ \texttt{j}^0\ \texttt{m}^0 &\equiv \texttt{if}_1\ \texttt{(leq m j)}\\
&\qquad\qquad \texttt{(k}_{00}\ \texttt{false)}\ (\lceil\texttt{sat-all G lam}\rceil)\\
&\qquad\qquad \texttt{(lam m)}
\end{aligned}$$

We achieve the interleaving of the first and second searches with the following, which applies both tests to the same index m:

$$\texttt{test-index G}^{k-1}\ \texttt{lam}^1\ \texttt{x}^0\ \texttt{m}^0 \equiv \texttt{or (refutes G x m)}$$

$$\begin{array}{rcl} & & (\lceil \texttt{settles G lam} \rceil \\ & & \quad (\texttt{position lam (proj0 x)})\ \texttt{m}) \\ \texttt{pre-critical-index}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{x}^0 & \equiv & \texttt{min}\ (\lceil \texttt{test-index G lam} \rceil\ \texttt{x}) \end{array}$$

If the index m discovered by the above is a refuter for G and x, we return m as the critical index. Otherwise, the token $\lambda(m)$ settles the value of $\Phi(G)$. If this value agrees with the value q given by the current element x, we wish to return a critical index that is *not* a refuter for G and x, so we may again return m. If the value disagrees with q, by restarting the first search we will eventually find a refuter which we then return.

$$\begin{array}{rcl} \texttt{test-pre-critical}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{x}^0\ \texttt{m}^0 & \equiv & \texttt{if}_0\ (\texttt{or}\ (\texttt{refutes G x m}) \\ & & \quad (\texttt{eq}\ (\texttt{proj2 x})\ (\texttt{proj2}\ (\texttt{lam m})))) \\ & & \texttt{m}\quad (\texttt{min}\ (\texttt{refutes G x})) \end{array}$$

We may now put the pieces together. Note that the element x in question is retrieved by looking up the given marker e in λ.

$$\begin{array}{rcl} \texttt{find-critical}\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{x}^0 & \equiv & \texttt{test-pre-critical G lam x} \\ & & \quad (\lceil \texttt{pre-critical-index G lam} \rceil\ \texttt{x}) \\ \texttt{critical-index}^*\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{e}^0 & \equiv & \texttt{find-critical G lam}\ (\texttt{lookup lam e}) \end{array}$$

The asterisk signals that this is not the official definition of `critical-index`. We next show how to eliminate the circularity from the above definition. The procedure is similar to that in Section 7.5, except that here the argument `lam` is considered as fixed, and the recursion involves only the parameter `e`.

We will introduce variants of the above definitions which carry around an auxiliary parameter $\texttt{aux} : \overline{1}$; this should be thought of as a candidate for `critical-index G lam`, which we are defining recursively. (Note that $\dot{G}$ and $\dot{\lambda}$ stay fixed throughout the computation, and do not actively participate in the recursion.) We start by defining

$$\texttt{critical-index}'\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{aux}^1 \quad \equiv \quad \texttt{aux}$$

We now perform the following for all defined constants dependent on `critical-index`, in the order of definition. Suppose $P : \overline{k-1} \to \overline{1} \to \tau$ is a constant which we defined earlier by means of an equation

$$P\ \texttt{G}^{k-1}\ \texttt{lam}^1\ x_0\ \cdots x_{r-1} \quad \equiv \quad E[\ \lceil Q\ \texttt{G lam} \rceil\ ]$$

where $E[-]$ abstracts the unique occurrence of a protected subexpression on the right hand side of the definition of P, and $Q : \overline{k-1} \to \overline{1} \to \sigma$ is some constant. Suppose moreover that we have already defined a variant $Q' : \overline{k-1} \to \overline{1} \to \overline{1} \to \sigma$. We may then define the variant $P' : \overline{k-1} \to \overline{1} \to \overline{1} \to \tau$ by means of the equation

$$P\ \texttt{G}^{k-1}\ \texttt{lam}^1\ \texttt{aux}^1\ x_0\ \cdots x_{r-1} \quad \equiv \quad E[\ \lceil Q'\ \texttt{G lam aux} \rceil\ ]$$

We apply this procedure in turn to the definitions of

`critical-index-thunk`, `interpret`, `G-value`, `satisfies`, `sat`, `sat-all`, `settles`, `test-index`, `pre-critical-index`, `find-critical`

Note that for each of these constants P, the definition given earlier has $\mathtt{G}^{k-1}$ and $\mathtt{lam}^1$ as the first two formal parameters, and at least one other parameter x_0. Moreover, the definition involves only one occurrence of a previously defined constant Q dependent on `critical-index`, and this appears as part of a protected subexpression $\lceil Q\ \mathtt{G}\ \mathtt{lam}\rceil$.

Finally, we may define

$$\begin{aligned}\mathtt{critical\text{-}index''}\ \mathtt{G}^{k-1}\ \mathtt{lam}^1\ \mathtt{aux}^1\ \mathtt{e}^0 \quad &\equiv \quad \mathtt{find\text{-}critical'}\ \mathtt{G}\ \mathtt{lam}\ \mathtt{aux}\\ &\qquad\qquad (\mathtt{lookup}\ \mathtt{lam}\ \mathtt{e})\\ \mathtt{critical\text{-}index}\ \mathtt{G}^{k-1}\ \mathtt{lam}^1 \quad &\equiv \quad \mathtt{y}_1\ (\mathtt{critical\text{-}index''}\ \mathtt{G}\ \mathtt{lam})\end{aligned}$$

This completes the definition of `critical-index`, and the definitions given earlier for all the non-primed constants listed above may now be allowed to stand as the official definitions of these constants. Our definition of $\mathtt{nabla}_k$ itself is thus complete, and we may take $\nabla_k \in A_{\overline{1}\to\overline{k}}$ to be the element defined by $\mathtt{nabla}_k$.

8.5. *Correctness of the Normann program*

It remains to show that ∇_k possesses the property required for Lemma 8.1. First, we have the following useful property for `find-critical`:

Lemma 8.12. For any $e \in \mathbb{N}$, $\dot\mu \in A_1$ and $\dot G \in A_{k-1}$ we have

$$\mathit{critical\text{-}index}\cdot\dot G\cdot\dot\lambda\cdot\widehat{e} \ \succeq\ \mathit{find\text{-}critical}\cdot\dot G\cdot\dot\lambda\cdot(\mathit{lookup}\cdot\dot\lambda\cdot\widehat{e})$$

Proof. Analogous to the proof of Lemma 7.11, setting $\mathit{aux} = \mathbf{y}_1\cdot(\mathit{critical\text{-}index''}\cdot\dot G\cdot\dot\lambda)$ and carrying the additional parameter $\dot\lambda$ through the argument. □

The main correctness proof is broadly similar to that of Proposition 7.12, with a few significant variations arising from the differing treatment of satisfactory representations.

Proposition 8.13. Suppose $\Phi \in \mathsf{HEO}_k$, $\nu \vdash^\delta_k \Phi$, $\dot\nu \in \mathsf{Tot}(A_1)$ represents ν, $G \in \mathsf{HEO}_{k-1}$ and $\dot G \in \mathsf{Tot}(A_{k-1})$ represents G. Then $\nabla_k\cdot\dot\nu\cdot G = \widehat{\Phi(G)}$.

Proof. Let $\dot\lambda = \mathit{satis\text{-}rep}\cdot\dot\nu$, so that $\dot\lambda$ represents the satisfactory representation $\lambda = \lambda^\nu$ of Φ.

By condition (4) of Definition 8.9, there is a position index j_0 such that $\lambda(j_0)$ has the form $x = \langle e, s, q\rangle$, where

$$s \ =\ \langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle$$

is a type $k-1$ signifier, and each $\langle\langle d_i, u_i, p_i\rangle\rangle$ is satisfactory for $\dot G$ relative to $\dot\lambda, e$. By condition (2) of Proposition 8.5, for each i we have $G(h) = p_i$ for all $h \in U_{[\langle d_i,u_i\rangle]}$, so by Proposition 4.10 we have $G \in U_{[\![\, s\,]\!]}$. So by condition (3) of Definition 8.9, we have $q = \Phi(G)$. Moreover, by condition (1) of Proposition 8.5, we have $\mathit{satisfies}\cdot\dot G\cdot\dot\lambda\cdot\widehat{e}\cdot\widehat{p_i}\cdot\widehat{d_i}\cdot\widehat{u_i} = \mathit{tt}$ for each i, and it follows easily that $\mathit{sat\text{-}all}\cdot\dot G\cdot\dot\lambda\cdot\widehat{x} \in \mathcal{T}$.

Set $x_j = \langle e_j, s_j, q_j\rangle = \lambda(j)$ for each j, so that $x_{j_0} = x$. We will show that the following hold for all $j \le j_0$:

(1) $\mathit{sat\text{-}all}\cdot\dot G\cdot\dot\lambda\cdot\widehat{x_j} \in \mathcal{T}\cup\mathcal{F}$.

(2) If $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_j} \in \mathcal{T}$, then $q_j = \Phi(G)$.

We argue by reversed induction on j.

For the case $j = j_0$, both claims are established by the above. So suppose $j < j_0$, and assume both claims hold for all j' where $j < j' \leq j_0$. Then it is clear that $\mathit{settles} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{j} \cdot \widehat{m} \in \mathcal{T} \cup \mathcal{F}$ for all $m \leq j_0$, and that $\mathit{settles} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{j} \cdot \widehat{j_0} \in \mathcal{T}$. Moreover, each x_j is a well-formed token, so by Proposition 8.11(i) we have that $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x_j} \cdot \widehat{m} \in \mathcal{T} \cup \mathcal{F}$ for every $m \in \mathbb{N}$. Since $\mathit{position} \cdot \dot{\lambda} \cdot (\mathit{proj0} \cdot \widehat{x_j}) = j$ it follows that $\mathit{pre\text{-}critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_j}$ successfully evaluates to some numeral $\widehat{m_0}$ where $m_0 \leq j_0$.

We next claim that $\mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{m_0}$ successfully evaluates to a numeral. Clearly, $\mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{m_0} = \widehat{m_0}$ unless we are in the case where $j < m_0 \leq j_0$ and $\lambda(m_0)$ settles $\Phi(G)$ (that is, $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_{m_0}} \in \mathcal{T}$) but $q_j \neq q_{m_0}$. In this latter case, by claim (2) of the induction hypothesis we have that $q_{m_0} = \Phi(G)$, whence $q_j \neq \Phi(G)$. But by condition (3) of Definition 8.9, $\langle e_j, s_j, q_j \rangle$ is consistent with Φ, so G cannot be consistent with $[\![s_j]\!]$. So by Proposition 8.11, there must be some smallest m_1 such that $\mathit{refutes} \cdot \dot{G} \cdot \widehat{x_j} \cdot m_1 \in \mathcal{T}$, and in this case, $\mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{m_0} = \widehat{m_1}$.

Using Lemma 8.12 and the remarks following the definition of `lookup`, we have that

$$\begin{aligned} \mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e_j} &\succeq \mathit{find\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_j} \\ &= \mathit{test\text{-}pre\text{-}critical} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{m_0} \end{aligned}$$

so $\mathit{critical\text{-}index} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e_j}$ evaluates to a numeral, $\widehat{m}$ say. By Proposition 8.3(ii), it follows that $\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{\langle d_i, u_i \rangle}$ is a total element of A_{k-2} for each $\langle d_i, u_i, p_i \rangle$ appearing in s_j, and hence that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_j} \in \mathcal{T} \cup \mathcal{F}$. This establishes claim (1) for j.

For claim (2), if $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_j} \in \mathcal{T}$ then we must have $\dot{G} \cdot (\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{\langle d_i, u_i \rangle}) = \widehat{p_i}$ for each $\langle d_i, u_i, p_i \rangle = t_i$ appearing in s_j. But $\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{\langle d_i, u_i \rangle} = \mathit{std\text{-}total} \cdot \widehat{d_i} \cdot \widehat{u_i} \cdot \widehat{m}$, so $\mathit{std\text{-}satisfies} \cdot \dot{G} \cdot \widehat{m} \cdot \widehat{t_i} = \mathit{tt}$ for each i. Hence

$$\mathit{refutes} \cdot \dot{G} \cdot \widehat{x_j} \cdot \widehat{m} = \mathit{ff}$$

So by inspection of the definition of `test-pre-critical`, we are in the case where $j < m \leq j_0$, $q_m = q_j$ and $\mathit{settles} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{j} \cdot \widehat{m} \in \mathcal{T}$. By the definition of `settles`, this means that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_m} \in \mathcal{T}$, and by claim (2) of the induction hypothesis, this implies that $q_m = \Phi(G)$. Hence $q_j = \Phi(G)$ as required.

To complete the proof, let j_1 be the least j such that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_j} \in \mathcal{T}$ (note that $j_1 \leq j_0$). Then by claim (1) above, we have that $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{x_j} \in \mathcal{F}$ for all $j < j_1$; hence $\mathit{search} \cdot \dot{\lambda} \cdot \dot{G} = \widehat{j_1}$. But now by claim (2) above we have $\nabla_k \cdot \dot{\nu} \cdot \dot{G} = \mathit{proj2} \cdot \widehat{x_j} = \widehat{\Phi(G)}$ and we are done. □

This completes the proof of Theorem 5.12.

9. The modified extensional collapse

As we saw in Section 4, the extensional collapse construction gives us a simple and uniform way of extracting a total type structure from a general N-TPCA. However, there is a second such construction which has proved to be of interest in the study of realizability

models, and which we call the *modified extensional collapse*. This arises naturally in connection with Kreisel's modified realizability interpretation and the associated categorical models (see (van Oosten 1997) for more information).

Rather pleasingly, it turns out that the proofs in Sections 7 and 8 can be adapted to yield analogous results for the modified extensional collapse construction. (This answers a question we posed in (Longley 2005b).) The purpose of this section is to present these adaptations. We will not give complete self-contained proofs here, but rather explain the points at which the earlier proofs need to be modified.

We first review the concrete definition of the modified extensional collapse.

In previous parts of the paper we have for brevity referred to the total, extensional elements of A simply as the "total elements", but for the purpose of this section we need to distinguish between the concepts of totality and extensionality.

Given any N-TPCA A, we may define subsets $A^{\mathsf{T}}_\sigma \subseteq A_\sigma$ as follows:

$$\begin{aligned} A^{\mathsf{T}}_0 &= \{\widehat{n} \mid n \in \mathbb{N}\} \\ A^{\mathsf{T}}_{\sigma\to\tau} &= \{f \in A_{\sigma\to\tau} \mid \forall x \in A^{\mathsf{T}}_\sigma.\ f\cdot x\downarrow \wedge f\cdot x \in A^{\mathsf{T}}_\tau\} \end{aligned}$$

It is easy to see that this defines a sub-N-TPCA A^{T} of A in which the application operations are total. For the purpose of this section, we will refer to the elements of A^{T} as the *total* elements of A.

Once we have constructed A^{T}, we may of course take its extensional collapse $\mathsf{EC}(A^{\mathsf{T}})$, and we refer to this as the *modified extensional collapse* of A, written $\mathsf{MEC}(A)$. We also write $\mathbf{MEC}(A)$ for the evident internal TTS within $\mathbf{Asm}(A)$ arising from this construction. For the purpose of this section, the elements in the field of the corresponding partial equivalence relations will be called the *extensional elements of* A; we write $\mathsf{Ext}(A_\sigma)$ for the set of such elements at type σ, so that $\mathsf{Ext}(A_\sigma) \subseteq A^{\mathsf{T}}_\sigma$. Note, however, that in connection with P and $\mathsf{P}^{\mathit{eff}}$ we shall still frequently need to refer to the sets $\mathsf{Tot}(\mathsf{P}_\sigma)$ and $\mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_\sigma)$ as defined previously.

We may regard the total elements of A_σ as "potential realizers" for functionals of type σ, and the extensional elements as "actual realizers". At types 0 and 1, every potential realizer is also an actual realizer. At type 2, the actual realizers are the same as for the standard realizability construction (hence $\mathbf{MEC}(A)_2 = \mathbf{EC}(A)_2$), but the set of potential realizers is in general larger, since typically there are total type 2 elements that do not act extensionally on type 1 actual realizers. This means that the set of type 3 functionals might be smaller in $\mathsf{MEC}(A)$ than in $\mathsf{EC}(A)$, since actual realizers at type 3 must not only map actual realizers to actual realizers, but also potential realizers to potential realizers. At type 4 and above the two TTSs may in general become incomparable.

In some particular instances (such as P) it is fairly easy to show that $\mathsf{EC}(A)$ and $\mathsf{MEC}(A)$ coincide. In other instances this can be shown by means of a non-trivial theorem — for example, the isomorphisms $\mathsf{EC}(K_1) \cong \mathsf{MEC}(K_1)$ and $\mathsf{EC}(K_2) \cong \mathsf{MEC}(K_2)$ were established in (Bezem 1985).

Our previous theorems as they stand do not apply to A^{T}, since it does not possess $\mathbf{y}$ combinators. However, in both the continuous and effective settings, we are able to obtain analogous theorems which identify $\mathsf{MEC}(A)$ in a wide range of instances. In each

case, we use the **y** combinators from A to construct the appropriate Normann programs, but some further refinements are needed to ensure that these programs return elements in A^{T}.

9.1. *The continuous case*

In the continuous case, our theorem for the modified extensional collapse is precisely analogous to Theorem 5.13:

Theorem 9.1. Suppose A is a full continuous NR-TPCA (with $\gamma : A \longrightarrow\!\!\triangleright K_2$) satisfying conditions (AB) and (C) of Theorem 5.13. Then $\gamma_*(\mathbf{MEC}(A)) \cong \mathbf{C}$ in $\mathbf{Asm}(K_2)$.

The proof of this theorem is closely parallel to that of Theorem 5.13. We here go through the proof in outline, elaborating on the points that are different.

As in Section 7 we formulate a lemma asserting the existence of suitable Normann programs, now understanding the phrase "$\dot{x}$ *represents* $x \in \mathsf{C}_l$" to mean that $\dot{x}$ is an element of $\mathsf{Ext}(A_l)$ whose action corresponds to that of x. Here, as in Section 7, δ denotes the realization $\mathsf{C} \longrightarrow\!\!\triangleright K_2$ via basic enumerations.

Lemma 9.2. Suppose $k \geq 1$, and suppose $\gamma_*(\mathbf{MEC}(A))$ and $\mathbf{C}$ are isomorphic up to level $k-1$. Then there is an NRComb$^+$-definable constant $\mathtt{nabla}_k : \overline{1} \to \overline{k}$ denoting an element $\nabla_k \in A_{\overline{1}\to\overline{k}}$ with the following property: whenever $\Phi \in \mathsf{C}_k$, $\nu \Vdash^{\delta}_k \Phi$ and $\dot{\nu} \in \mathsf{Ext}(A_1)$ represents ν, the element $\nabla_k \cdot \dot{\nu} \in A_k$ represents Φ within A (in the modified sense).

The proof that this lemma implies the theorem then goes through as for Theorem 7.2 with only minor changes. Moreover, the proofs of the lemma for types 1 and 2 are unaffected, since $\mathbf{MEC}(A)_1 = \mathbf{EC}(A)_1$ and $\mathbf{MEC}(A)_2 = \mathbf{EC}(A)_2$.

For a general type level $k \geq 3$, we assume the lemma holds for $k-2$, and adopt the machinery of Section 7.2 as it stands, except that we insert a normalization into the definition of `partial-elt`:

$$\mathtt{partial\text{-}elt}\ \mathtt{mu}^1\ \mathtt{thunk}^1 \quad \equiv \quad \mathtt{nabla}_{k-2}\ (\mathtt{norm}\ (\mathtt{extend\text{-}enum}\ \mathtt{mu}\ \mathtt{thunk}))$$

This enables us to show, for instance, that if θ is a basic enumeration of some $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-1})$ then

$$\mathit{interpret} \cdot \dot{G} \cdot \dot{\lambda} \cdot \mu^{\theta\sharp} \ = \ \nabla_{k-2} \cdot \theta^{\sharp}$$

for any $\dot{\lambda} \in A_1$ (where $f^{\sharp}$ denotes the canonical representative within A_1 of $f : \mathbb{N} \to \mathbb{N}$).

The first interesting issue concerns the crucial Proposition 7.6. The proposition as it stands still applies when $\dot{G}$ is extensional, but the same proof also yields some information for total but non-extensional $\dot{G}$:

Proposition 9.3. Suppose given $\dot{G} \in A^{\mathsf{T}}_{k-1}$, $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing $h \in \mathsf{C}_{k-2}$, θ a basic enumeration of $\mathfrak{h}$, and $\lambda : \mathbb{N} \to \mathbb{N}$. Then $\dot{G} \cdot (\nabla_{k-2} \cdot \theta^{\sharp})$ is a numeral, say $\widehat{p}$, and there exist $b, z \in \mathbb{N}$ with $b = \langle b_0, \ldots, b_{t-1}\rangle$ and $\zeta_{k-2}(b) \sqsubseteq \mathfrak{h}$ such that

(1) for any $\lambda' \in \mathcal{B}(\lambda, z)$ and $\mu' \in \mathcal{B}(\mu^b, t)$, we have

$$\mathit{satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot (\mathbf{pair} \cdot \lambda'^{\sharp} \cdot \mu'^{\sharp}) \;=\; tt$$

(so in particular $\mathit{satisfies\text{-}code} \cdot \dot{G} \cdot \lambda'^{\sharp} \cdot \widehat{b} \cdot \widehat{p} = tt$);

(2) if $\dot{G} \in \mathsf{Ext}(A_{k-1})$ and $\dot{G}$ represents $G \in \mathsf{C}_{k-1}$, then for all $h' \in O_{k-2,b}$ we have $G(h') = p$.

Proof. By the main induction hypothesis, we have $\nabla_{k-2} \cdot \theta^{\sharp} \in \mathsf{Ext}(A_{k-2}) \subseteq A^{\mathsf{T}}_{k-2}$, so $\dot{G} \cdot (\nabla_{k-2} \cdot \theta^{\sharp})$ is a numeral, say $\widehat{p}$. Hence $\mathit{interpret} \cdot \dot{G} \cdot \lambda^{\sharp} \cdot \mu^{\theta\sharp} = \widehat{p}$, so that $\mathit{satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot (\mathbf{pair} \cdot \lambda^{\sharp} \cdot \mu^{\theta\sharp}) = tt$, even if $\dot{G}$ is non-extensional. The construction of suitable b, z, and the verification that they satisfy condition (1), then work exactly as in Proposition 7.6. For extensional $\dot{G}$, condition (2) of course holds as before. □

For any $\dot{G} \in A^{\mathsf{T}}_{k-1}$, we may say (b, p) is *satisfactory* for $\dot{G}$ relative to λ if $b \in \mathrm{dom}\ \zeta_{k-2}$ and there exists z such that b, p, z satisfy conditions (1) and (2) above (condition (2) being vacuously true if $\dot{G} \notin \mathsf{Ext}(A_{k-1})$). We write $S_\lambda(\dot{G})$ for the set of all satisfactory pairs for $\dot{G}$ relative to λ.

We will say an element $\dot{G} \in A^{\mathsf{T}}_{k-1}$ is *quasi-extensional* if whenever θ, θ' are (respectively) basic enumerations for elements $\mathfrak{h}, \mathfrak{h}' \in \mathsf{Tot}(\mathsf{P}_{k-2})$ both representing $h \in \mathsf{C}_{k-2}$, we have

$$\dot{G} \cdot (\nabla_{k-2} \cdot \theta^{\sharp}) \;=\; \dot{G} \cdot (\nabla_{k-2} \cdot \theta'^{\sharp})$$

Clearly, any extensional element is quasi-extensional. The following proposition gives some useful equivalents of quasi-extensionality. Here we write filter : $\mathbb{N}^2 \to \mathbb{N}$ for the function defined by the program `filter` from Section 7.2; as before we also define

$$\texttt{std-total}\ \mathtt{b}^0\ \mathtt{c}^0 \;\equiv\; \texttt{partial-elt}\ (\texttt{mod-enum}\ \mathtt{b})\ (\mathtt{k}\ \mathtt{c})$$

Proposition 9.4. The following are equivalent for any $\dot{G} \in A^{\mathsf{T}}_{k-1}$ and $\lambda \in \mathbb{N}^{\mathbb{N}}$:

(1) $\dot{G}$ is quasi-extensional.

(2) For all $b, c, b', c' \in \mathrm{dom}\ \zeta_{k-2}$, if $\mathrm{filter}(b, c)$ and $\mathrm{filter}(b', c')$ code the same type $k-2$ compact element then

$$\dot{G} \cdot (\mathit{std\text{-}total} \cdot \widehat{b} \cdot \widehat{c}) \;=\; \dot{G} \cdot (\mathit{std\text{-}total} \cdot \widehat{b'} \cdot \widehat{c'})$$

(3) The codes $\langle b \mapsto \breve{p} \rangle$ for $(b, p) \in S_\lambda(\dot{G})$ are pairwise consistent.

Proof. (1) ⇒ (2): For any $b, c \in \mathrm{dom}\ \zeta_{k-2}$ we have

$$\mathit{std\text{-}total} \cdot \widehat{b} \cdot \widehat{c} \;=\; \nabla_{k-2} \cdot (\mathit{norm} \cdot (\mathit{extend\text{-}enum} \cdot \widehat{b} \cdot (\mathbf{k} \cdot \widehat{c})))$$

But $\mathit{extend\text{-}enum} \cdot \widehat{b} \cdot (\mathbf{k} \cdot \widehat{c})$ clearly represents some basic enumeration θ for $\mathfrak{r}_{c^*}$, where $c^* = \mathrm{filter}(b, c)$; hence $\mathit{std\text{-}total} \cdot \widehat{b} \cdot \widehat{c} = \nabla_{k-2} \cdot \theta^{\sharp}$. We also have a similar scenario involving b', c', c'^*, θ'. So if $c^* = c'^*$ and $\dot{G}$ is quasi-extensional then $\nabla_{k-2} \cdot \theta^{\sharp} = \nabla_{k-2} \cdot \theta'^{\sharp}$, whence $\mathit{std\text{-}total} \cdot \widehat{b} \cdot \widehat{c} = \mathit{std\text{-}total} \cdot \widehat{b'} \cdot \widehat{c'}$.

(2) ⇒ (3): Suppose given $(b, p), (b', p')$ in $S_\lambda(\dot{G})$ where $b = \langle b_0, \ldots, b_{t-1} \rangle$ and $b' = \langle b'_0, \ldots, b'_{t'-1} \rangle$ are consistent. Let b'' be a code for $\zeta_{k-2}(b) \sqcup \zeta_{k-2}(b')$. Then $\mathit{extend\text{-}enum} \cdot$

$\widehat{b} \cdot (\mathbf{k} \cdot \widehat{b''})$ represents some $\theta \in \mathcal{B}(\mu^b, t)$ which basically enumerates $\mathfrak{x}_{b''}$, so

$$\textit{std-total} \cdot \widehat{b} \cdot \widehat{b''} \;=\; \nabla_{k-2} \cdot (\textit{norm} \cdot (\textit{extend-enum} \cdot \widehat{b} \cdot (\mathbf{k} \cdot \widehat{b''}))) \;=\; \nabla_{k-2} \cdot \theta^\sharp$$

Now $\dot{G} \cdot (\nabla_{k-2} \cdot \theta^\sharp)$ is certainly a numeral, and since (b, p) is satisfactory for $\dot{G}$ relative to λ it must be $\widehat{p}$, otherwise we would have

$$\begin{aligned} \textit{satisfies} \cdot \dot{G} \cdot \widehat{p} \cdot (\mathbf{pair} \cdot \lambda^\sharp \cdot \mu^{\theta\sharp}) &= \textit{eq} \cdot (\dot{G} \cdot (\textit{partial-elt} \cdot \mu^{\theta\sharp} \cdot \textit{thunk})) \cdot \widehat{p} \\ &= \textit{eq} \cdot (\dot{G} \cdot (\nabla_{k-2} \cdot \theta^\sharp)) \cdot \widehat{p} \;=\; \textit{ff} \end{aligned}$$

for a certain $\textit{thunk} \in A_1$. Thus $\dot{G} \cdot (\textit{std-total} \cdot \widehat{b} \cdot \widehat{b''}) = \widehat{p}$, and similarly $\dot{G} \cdot (\textit{std-total} \cdot \widehat{b'} \cdot \widehat{b''}) = \widehat{p'}$. But $\mathrm{filter}(b, b'') = \mathrm{filter}(b', b'') = b''$, so if condition (2) holds then $p = p'$.

(3) $\Rightarrow$ (1): Suppose θ, θ' are basic enumerations for $\mathfrak{h}, \mathfrak{h}'$ which both represent $h \in \mathsf{C}_{k-2}$, and let $\widehat{p} = \dot{G} \cdot (\nabla_{k-2} \cdot \theta^\sharp)$, $\widehat{p'} = \dot{G} \cdot (\nabla_{k-2} \cdot \theta'^\sharp)$. Then by Proposition 9.3 we may find b, b' such that $\zeta_{k-2}(b), \zeta_{k-2}(b') \sqsubseteq \mathfrak{h}$ and $(b, p), (b', p') \in S_\lambda(\dot{G})$. But b and b' are consistent, since $h \in O_{k-2,b} \cap O_{k-2,b'}$. So if condition (3) holds we must have $p = p'$. □

Suppose $\dot{G} \in A^{\mathsf{T}}_{k-1}$ and $\lambda \in \mathbb{N}^\mathbb{N}$. Then for any $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}_{k-2})$, by Proposition 9.3 there is some $(b, p) \in S_\lambda(\dot{G})$ such that $\zeta_{k-2}(b) \sqsubseteq \mathfrak{h}$. Thus, if $\dot{G}$ is quasi-extensional so that $S_\lambda(\dot{G})$ is consistent, we have that $\mathfrak{G}_\lambda = \bigsqcup\{\zeta_{k-1}(\langle b \mapsto \breve{p}\rangle) \mid (b, p) \in S_\lambda(\dot{G})\}$ is an element of $\mathsf{Tot}(\mathsf{P}_{k-1})$ and so represents some $G \in \mathsf{C}_{k-1}$. It is now easy to see that whenever θ basically enumerates some $\mathfrak{h}$ representing $h \in \mathsf{C}_{k-2}$ we have $\dot{G} \cdot (\nabla_{k-2} \cdot \theta^\sharp) \;=\; \widehat{G(h)}$; in this case we say that $\dot{G}$ *quasi-represents* G.

Proposition 7.7 fails badly in the present setting, since in general the set of basic compacts obtained from $S_\lambda(\dot{G})$ will not be consistent. However, salvaging what we can from the situation, we arrive at the following useful dichotomy: *either* $\dot{G}$ is quasi-extensional, in which case $\mathfrak{G}_\lambda$ represents some $G \in \mathsf{C}_{k-1}$ which is also quasi-represented by $\dot{G}$; *or* there exist $(b, p), (b', p') \in S_\lambda(\dot{G})$ such that the codes $\langle b \mapsto \breve{p}\rangle$, $\langle b' \mapsto \breve{p'}\rangle$ are inconsistent.

It can now be seen that the Normann program given in Section 7 works as desired when $\dot{G}$ is quasi-extensional. More precisely, if $\Phi \in \mathsf{C}_k$, $\nu \Vdash^\delta_k \Phi$, $\dot{\nu} \in \mathsf{Ext}(A_1)$ represents ν and $\dot{G} \in \mathsf{Ext}(A_{k-1})$ quasi-represents $G \in \mathsf{C}_{k-1}$, then $\nabla_k \cdot \dot{\nu} \cdot \dot{G} = \widehat{\Phi(G)}$. So in order to achieve $\nabla_k \cdot \dot{\nu} \in \mathsf{Ext}(A_k)$, it remains to modify the Normann programs to ensure that, additionally, $\nabla_k \cdot \dot{\nu} \cdot \dot{G}$ evaluates to some numeral (it does not matter which) when $\dot{G}$ is total but not quasi-extensional.

We may achieve this quite easily by combining the existing Normann program with a search for two inconsistent satisfactory pairs witnessing that $\dot{G}$ is not quasi-extensional. That is, we build into our satisfactory representation not only the codes $c \mapsto \breve{q}$ arising from the enumeration ν, but also all possible "pseudo-codes" $\langle b \mapsto \breve{p}, b' \mapsto \breve{p'}\rangle \mapsto \breve{0}$ where $\langle b \mapsto \breve{p}\rangle, \langle b' \mapsto \breve{p'}\rangle$ are inconsistent:

Definition 9.5. Suppose $\Phi \in \mathsf{C}_k$. A *modified satisfactory representation* of Φ is a total function $\lambda : \mathbb{N} \to \mathbb{N}$ such that

(1) for all j, the number $\lambda(j)$ is either 0 or of the form $2(c \mapsto \breve{q}) + 1$ or $2(c \mapsto \breve{q}) + 2$, where either $c \in \mathrm{dom}\ \zeta_{k-1}$ and $q \in \mathbb{N}$, or c is of the form $\langle c_0, c_1\rangle$ where $\langle c_0\rangle, \langle c_1\rangle$ are inconsistent basic type $k-1$ codes and $q = 0$;

(2) whenever $\lambda(j) = 2(c \mapsto \breve{q}) + 1$, there exists $j' > j$ such that $\lambda(j'') = 0$ whenever $j < j'' < j'$, and $\lambda(j') = 2(c \mapsto \breve{q}) + 2$;
(3) whenever $\lambda(j) = 2(c \mapsto \breve{q}) + 1$, if c is a consistent ζ_{k-1}-code then the basic type k code $\langle c \mapsto q \rangle$ is consistent with Φ (that is, $\Phi \in O_{k,\,\langle c \mapsto q \rangle}$);
(4) for any $\dot{G} \in A^{\mathsf{T}}_{k-1}$, there exists j_0 such that $\lambda(j_0)$ has the form

$$2(\langle b_0 \mapsto \breve{p}_0, \ldots, b_{r-1} \mapsto \breve{p}_{r-1} \rangle \mapsto \breve{q}) + 1$$

where each pair (b_i, p_i) is satisfactory for $\dot{G}$ relative to λ^{j_0+1}.

Satisfactory representations of the new kind still exist and are computable from the basic enumerations ν. The construction is virtually identical to that in the proof of Proposition 7.9, except that we replace the condition "$\langle e \rangle \in \mathrm{range}(\nu')$" everywhere by "$\langle e \rangle \in \mathrm{range}(\nu')$ or $e = \langle c_0, c_1 \rangle \mapsto 0$ where $\langle c_0 \rangle, \langle c_1 \rangle$ are inconsistent ζ_{k-1}-codes". To see that condition (4) above is satisfied, we appeal to the dichotomy mentioned earlier: either $\dot{G}$ is quasi-extensional or there exist $(b, p), (b', p') \in S_{\lambda_0}(\dot{G})$ giving rise to inconsistent codes, where λ_0 is the constant zero function. In either case, an argument similar to the one in the proof of Proposition 7.9 shows that a suitable position index j_0 exists.

As before, we take $\texttt{satis-rep} : \overline{1} \to \overline{1}$ to be a program that computes a satisfactory representation λ^ν from a basic enumeration ν.

We now define our Normann programs as in Section 7.5. Only one further change to the code is needed: at a certain juncture we interleave our "first search" for a refuter for $\dot{G}$ and x with a search for a witness that $\dot{G}$ is "inconsistent" (*i.e.*, not quasi-extensional). We suppose $\texttt{same-compact} : \overline{0}^{(4)} \to \overline{0}$ represents a total computable function which, given $b_0, c_0, b_1, c_1 \in \mathbb{N}$, tests whether $\mathrm{filter}(b_0, c_0)$ and $\mathrm{filter}(b_1, c_1)$ code the same type $k-2$ compact element.

```
incons4 G^{k-1} b0^0 c0^0 b1^0 c1^0  ≡  and (same-compact b0 c0 b1 c1)
                                           (not (eq (G (std-total b0 c0))
                                                    (G (std-total b1 c1))))
incons G^{k-1} m^0                   ≡  incons4 (proj0 m) (proj1 m)
                                           (proj2 m) (proj3 m)
refutes-or-incons G^{k-1} x^0 m^0    ≡  or (refutes G x m) (incons G m)
```

In the definition of `test-pre-critical`, we replace the second occurrence of `refutes` (only) by `refutes-or-incons`:

```
test-pre-critical G^{k-1} lam^1 x^0 m^0  ≡  if (or (refutes G x m)
                                                   (and (not (even (lam m)))
                                                        (eq (result x)
                                                            (result (lam m)))))
                                               m  (min (refutes-or-incons G x))
```

With these changes in place, we obtain a modified Normann program $\texttt{nabla}_k$ and the corresponding operator $\nabla_k \in A_{\overline{1} \to \overline{k}}$. It remains to verify that this does what is required:

Proposition 9.6. Suppose $\Phi \in \mathsf{C}_k$, $\nu \Vdash_k^\delta \Phi$, $\dot{\nu} \in \mathsf{Ext}(A_1)$ represents ν and $\dot{G} \in A_{k-1}^{\mathsf{T}}$. Then

(1) $\nabla_k \cdot \dot{\nu} \cdot \dot{G}$ evaluates to a numeral;
(2) if $\dot{G}$ quasi-represents $G \in \mathsf{C}_{k-1}$, then $\nabla_k \cdot \dot{\nu} \cdot \dot{G} = \widehat{\Phi(G)}$.

The formal proof is a mild elaboration of the proof of Proposition 7.12. We content ourselves here with some informal remarks. It is convenient to consider separately the cases where $\dot{G}$ is quasi-extensional and where $\dot{G}$ is inconsistent.

In the former case, the proof of Proposition 7.12 goes through smoothly (we note that Proposition 7.10 holds under the weaker hypothesis that $\dot{G} \in A_{k-1}^{\mathsf{T}}$ quasi-represents G). The "inconsistent codes" in the satisfactory representation do not get in the way of the computation, since a quasi-extensional $\dot{G}$ will never be found to satisfy both of the corresponding tests (except in cases where $\Phi(G)$ is known to be 0 anyway). Likewise, the tests involving `incons` do not get in the way, since these tests will always fail.[†]

In the latter case, by condition (4) of Definition 9.5 there will be some inconsistent code $\lambda(j_0)$ in the satisfactory representation that will serve as a suitable "end-stop", so we merely have to show by reversed induction that the tests $\mathit{sat\text{-}all} \cdot \dot{G} \cdot \dot{\lambda}^{j+1} \cdot \widehat{x_j}$ for $j \leq j_0$ all successfully evaluate to some value in $\mathcal{T} \cup \mathcal{F}$. For this, it suffices to show that $\mathit{critical\text{-}index} \cdot \dot{G} \cdot \lambda^{j+1}$ successfully evaluates to a numeral. But the search embodied by `pre-critical-index` will terminate as it is bounded by the end-stop j_0, whilst the search using `refutes-or-incons` will terminate since a quadruple (b_0, c_0, b_1, c_1) witnessing the inconsistency certainly exists.

Proposition 9.6 now immediately implies that if $\nu \Vdash_k^\delta \Phi \in \mathsf{C}_k$ then $\nabla_k \cdot \dot{\nu} \in \mathsf{MEC}_k$ and $\nabla_k \cdot \dot{\nu}$ represents Φ. This completes the proof of Theorem 9.1.

9.2. *The effective case*

For the effective case, we have a theorem analogous to Theorem 5.12. Here it turns out that a slight strengthening of condition (B) is helpful.[‡]

Theorem 9.7. Let A be an effective NR-TPCA, with γ a realization of A over K_1 satisfying the following conditions:

(A) If $m \vdash^\gamma_{\sigma \to \overline{0}} a$, $n \vdash^\gamma_\sigma b$ and $\mathbf{a}_{\sigma 0} \bullet m \bullet n \vdash^\gamma_0 c$, then $a \cdot b = c$.
(B′) The relation "$m \vdash^\gamma_0 \widehat{n}$" is c.e. in m, n.

Then $\gamma_*(\mathbf{MEC}(A)) \cong \mathbf{HEO}$ in $\mathbf{Asm}(K_1)$.

Again the proof closely follows that of Theorem 5.12, and the adaptations that are needed are rather similar to those used in the continuous case. However, the task of

[†] Somewhat more formally, under the hypothesis that $\dot{G} \in A_{k-1}^{\mathsf{T}}$ quasi-represents G, the proof of Proposition 7.12 goes through line by line, except that at the point where we invoke condition (3) of the definition of satisfactory representation, we need to say: either $c_j \mapsto \breve{q}_j$ or c_j is itself inconsistent, so in either case, G cannot be consistent with all the $b_i \mapsto \breve{p}_i$ appearing in c_j.

[‡] It seems likely that our condition (B') is strictly stronger than (B), although we do not have an example to prove this.

dealing correctly with total but non-extensional realizers in the absence of condition (C) calls for rather more delicate arguments than any of the preceding proofs.

The early part of the proof (up to the start of Section 8.1) goes through with only bureaucratic changes. For convenience, we state here the modified version of the main lemma. Note that δ here denotes the standard realization $\mathsf{HEO} \longrightarrow\!\!\!\!\triangleright K_2^{\mathit{eff}}$ via basic enumerations, and as in Section 9.1, we understand the phrase "$\dot{x} \in A_l$ *represents* x" to mean that $\dot{x}$ is an element of $\mathsf{Ext}(A_l)$ whose action corresponds to that of x.

Lemma 9.8. Suppose $k \geq 1$, and suppose $\gamma_*(\mathbf{MEC}(A))$ and **HEO** are isomorphic up to level $k-1$. Then there is an NRComb-definable constant $\mathtt{nabla}_k : \overline{1} \to \overline{k}$ denoting an element $\nabla_k \in A_{\overline{1}\to\overline{k}}$ with the following property: whenever $\Phi \in \mathsf{HEO}_k$, $\nu \Vdash_k^\delta \Phi$ and $\dot{\nu} \in \mathsf{Ext}(A_1)$ represents ν, the element $\nabla_k \cdot \dot{\nu} \in A_k$ represents Φ within A (in the modified sense).

As before, our task is to prove this lemma for $k \geq 3$, assuming that it holds for $k-2$.

We import the material in Section 8.1 unchanged. In contrast to the modified continuous case, there is no need for a normalization in the definition of `partial-elt`.

$$\mathtt{partial\text{-}elt}\ \mathtt{z}^0\ \mathtt{thunk}^1 \quad \equiv \quad \mathtt{nabla}_{k-2}\ (\mathtt{extend\text{-}enum}\ \mathtt{z}\ \mathtt{thunk})$$

We also define

$$\mathtt{total\text{-}elt}\ \mathtt{d}^0 \quad \equiv \quad \mathtt{nabla}_{k-2}\ (\mathtt{bullet}\ \mathtt{d})$$

so that if d is a Kleene index for a basic enumeration of a representative of some $h \in \mathsf{HEO}_{k-2}$ then $\mathit{total\text{-}elt} \cdot \widehat{d}$ represents h.

As in Section 9.1, all goes smoothly until we reach the crucial continuity property (Propositions 8.4 and 8.5). At this point some further notation is helpful. As in Section 8.2, we take

$$\begin{aligned} J \;=\; \{(d,v) \;\mid\; & \mathit{time}(v) < \infty \;\wedge\; \forall i < \mathit{time}(v).\ d \bullet i \in \mathrm{dom}\ \zeta_{k-2} \;\wedge \\ & \forall i,j < \mathit{time}(v).\ \zeta_{k-2}(d \bullet i),\ \zeta_{k-2}(d \bullet j) \text{ are consistent}\} \end{aligned}$$

We will define a family of basic enumerations to play the role of the functions $\eta_{d,v}$ from Section 8.2, although here we require a larger class of functions so as to yield representatives of *all* the standard total elements $\mathfrak{x}_c$ compatible with a given $(d,v) \in J$.

Let $b(d,t)$ denote a ζ_{k-2}-code for $\bigsqcup_{i<t} \zeta_{k-2}(d \bullet i)$ (when this exists) effectively computed from d,t in some standard way. For any $d,v,m \in \mathbb{N}$, define $\eta_{d,v,m} : \mathbb{N} \rightharpoonup \mathbb{N}$ by

$$\eta_{d,v,m}(i) \;\simeq\; \begin{cases} d \bullet i & \text{if } i < \mathit{time}(v) \\ \xi_c(i) & \text{if } i \geq \mathit{time}(v) \text{ and } c = \mathrm{filter}(b(d, \mathit{time}(v)), m) \end{cases}$$

Note that $\eta_{d,v,m}$ is represented in A_1 (in the sense of Section 2.2) by $\mathit{extend\text{-}enum} \cdot \widehat{\langle d,v \rangle} \cdot (\mathbf{k} \cdot \widehat{m})$. Let $\delta(d,v,m)$ denote a Kleene index for $\eta_{d,v,m}$ obtained effectively from d,v,m. Now let

$$M \;=\; \{(d,v,m) \;\mid\; (d,v) \in J,\ m \in \mathbb{N}\}$$

Then M is c.e., and for any $(d,v,m) \in M$, $\eta_{d,v,m}$ is a basic enumeration for some element $\mathfrak{x}_c \in \mathsf{Tot}(\mathsf{P}_{k-2})$ representing an element of $O_{k-2,b(d,\mathit{time}(v))}$. For the purpose of

this section, define

$$D \;=\; \{\delta(d,v,m) \;\mid\; (d,v,m) \in M\}$$

and note that D is c.e.

We may now formulate a modified version of Proposition 8.4:

Proposition 9.9. Suppose given $\dot{G} \in A_{k-1}^{\mathsf{T}}$ and $d \in \mathbb{N}$ such that ϕ_d basically enumerates some $\mathfrak{h}$ representing $h \in \mathsf{HEO}_{k-2}$. Suppose also that $\dot{h} = \mathit{total\text{-}elt} \cdot \widehat{d}$ and $\dot{G} \cdot \dot{h} = \widehat{p}$. Then there exists a number $u \in H$ which *signifies a modified KLS modulus of* $\dot{G}$ *at* d, p in the following sense: if $\dot{G} \in \mathsf{Ext}(A_{k-1})$ and $\mathfrak{b} = \bigsqcup_{i<time(u)} \zeta_{k-2}(d \bullet i)$, then for all $\dot{h}' \in \mathsf{Ext}(A_{k-2})$ representing $h' \in U_{\mathfrak{b}}^{\mathsf{HEO}}$ we have that $\dot{G} \cdot \dot{h}' = \widehat{p}$. (If $\dot{G}$ is non-extensional, any $u \in H$ is vacuously considered to signify a modified KLS of $\dot{G}$ at d, p.)

Moreover, a suitable value for u may be effectively computed from d and any realizer $y \vdash_{k-1}^{\gamma} \dot{G}$. In particular, there is an effective procedure for computing u from d and y which yields some number $u \in H$ even if $\dot{G}$ is non-extensional.

Proof. The construction of u used in the proof of Proposition 8.4 unfortunately does not guarantee termination for all $y \vdash^{\gamma} \dot{G} \in A_{k-1}^{\mathsf{T}}$. However, a more delicate construction may be given by adapting Gandy's proof of the KLS theorem (Gandy 1962), which involves a seemingly magical appeal to Kleene's second recursion theorem. Similar arguments also appear in (Bezem 1985).

We start by importing the definition of `easy-critical-index` from the proof of Proposition 8.4, with one subtle change to the definition of `std-dense`. (This change is not required for the proof of the present proposition, but is needed for the proof of Proposition 9.13 below.) Let $\mathtt{delta} : \overline{0}^{(3)} \to \overline{0}$ be a program that uniformly represents the function δ introduced above, and define

$$\begin{aligned}
\mathtt{std\text{-}dense}\ \mathtt{z}^0\ \mathtt{m}^0 \;&\equiv\; \mathtt{total\text{-}elt}\ (\mathtt{delta}\ (\mathtt{proj0}\ \mathtt{z})\ (\mathtt{proj1}\ \mathtt{z})\ \mathtt{m}) \\
\mathtt{is\text{-}counterex\text{-}for}\ \mathtt{G}^{k-1}\ \mathtt{z}^0\ \mathtt{p}^0\ \mathtt{m}^0 \;&\equiv\; \mathtt{not}\ (\mathtt{eq}\ (\mathtt{G}\ (\mathtt{std\text{-}dense}\ \mathtt{z}\ \mathtt{m}))\ \mathtt{p}) \\
\mathtt{easy\text{-}critical\text{-}index}\ \mathtt{G}^{k-1}\ \mathtt{z}^0\ \mathtt{p}^0 \;&\equiv\; \mathtt{min}\ (\mathtt{is\text{-}counterex\text{-}for}\ \mathtt{G}\ \mathtt{z}\ \mathtt{p})
\end{aligned}$$

We will also need some notation for operations on realizers that mirror various operations representable in A; these are constructed using the realizers $\mathbf{a}_{\sigma\tau}$ associated with γ. First, let $\tau(-)$ be a total computable function such that $\tau(d) \vdash_1^{\gamma} \mathit{total\text{-}elt} \cdot \widehat{d}$ for any $d \in \mathbb{N}$. Next, using conditions (A) and (B′) of Theorem 9.7, we may define a partial computable function $\mathit{eval}(-,-)$ such that, whenever $\dot{G} \in A_{k-1}$, $\dot{h} \in A_{k-2}$, $y \vdash_{k-1}^{\gamma} \dot{G}$ and $x \vdash_{k-2}^{\gamma} \dot{h}$, we have $\mathit{eval}(y,x) = p$ iff $\dot{G} \cdot \dot{h} = \widehat{p}$. We write $\mathit{evaltime}(y,x)$ for the number of steps taken to compute $\mathit{eval}(y,x)$ on some fixed Turing machine (we set $\mathit{evaltime}(y,x) = \infty$ if the computation diverges). Likewise, we may construct a partial computable function $\iota(-,-,-,-)$ such that whenever $\dot{G} \in A_{k-1}$, $y \vdash^{\gamma} \dot{G}$ and $d, u, p \in \mathbb{N}$, we have $\iota(y,d,u,p) = m$ iff $\mathit{easy\text{-}critical\text{-}index} \cdot \dot{G} \cdot \widehat{\langle d,u \rangle} \cdot \widehat{p} = \widehat{m}$. As one further piece of notation, we suppose we have a computable function $u(-)$ such that $\mathit{time}(u(t)) = t$ for any t (it is harmless to insist that $\mathit{time}(-)$ is defined in such a way that some such function exists).

We now construct a "critical" Kleene index $a_{y,d}$ with certain self-referential properties.

The construction involves a use of recursion, for which we temporarily introduce an auxiliary parameter a.

For any $y, d, a \in \mathbb{N}$, define a partial computable function $\psi_{y,d,a} : \mathbb{N} \rightharpoonup \mathbb{N}$ by

$$\psi_{y,d,a}(i) \simeq \begin{cases} d \bullet i & \text{if } i < \mathit{evaltime}(y, \tau(a)) \\ d \bullet i & \text{if } i \geq \mathit{evaltime}(y, \tau(a)) \text{ and } \mathit{eval}(y, \tau(a)) \neq \mathit{eval}(y, \tau(d)) \\ \xi_{k-2,c}(i) & \text{if } i \geq \mathit{evaltime}(y, \tau(a)) = t, \\ & \mathit{eval}(y, \tau(a)) = \mathit{eval}(y, \tau(d)) = p, \\ & m = \iota(y, d, u(t), p) \text{ and } c = \mathrm{filter}(b(d, t), m) \end{cases}$$

(Note that the value of $\mathit{evaltime}(y, \tau(a))$ does not include any time taken to compute $\tau(a)$.) Now let $\mathbf{Y}$ denote some fixed point combinator within K_1 such that $\mathbf{Y} \bullet f \downarrow$ and $\mathbf{Y} \bullet f \bullet x \simeq f \bullet (\mathbf{Y} \bullet f) \bullet x$ for every $f, x \in \mathbb{N}$. For any y, d, define $a_{y,d} = \mathbf{Y} \bullet (\Psi \bullet y \bullet d)$, so that $a_{y,d} \bullet i \simeq \Psi \bullet y \bullet d \bullet a_{y,d} \bullet i$ for all i. The upshot of this is that we have an index $a_{y,d}$, computable in y, d, such that for all $i \in \mathbb{N}$ we have

$$a_{y,d} \bullet i \simeq \begin{cases} d \bullet i & \text{if } i < \mathit{evaltime}(y, \tau(a_{y,d})) \\ d \bullet i & \text{if } i \geq \mathit{evaltime}(y, \tau(a_{y,d})) \text{ and } \mathit{eval}(y, \tau(a_{y,d})) \neq \mathit{eval}(y, \tau(d)) \\ \xi_{k-2,c}(i) & \text{if } i \geq \mathit{evaltime}(y, \tau(a_{y,d})) = t, \\ & \mathit{eval}(y, \tau(a_{y,d})) = \mathit{eval}(y, \tau(d)) = p, \\ & m = \iota(y, d, u(t), p) \text{ and } c = \mathrm{filter}(b(d, t), m) \end{cases}$$

Now let $\dot{G} \in A_{k-1}^{\mathsf{T}}, y, d, p$ be as in the Proposition, so that $\mathit{eval}(y, \tau(d)) = p$. We claim that $\mathit{eval}(y, \tau(a_{y,d}))$ is defined. Suppose not; then $\mathit{evaltime}\ (y, \tau(a_{y,d})) = \infty$ so $a_{y,d} \bullet i = d \bullet i$ for all i, so $\mathit{bullet} \cdot \widehat{a_{y,d}}$ represents a basic enumeration of $\mathfrak{h}$. Hence $\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}$ is in $\mathsf{Ext}(A_{k-2})$ and represents h. Since $\dot{G} \in A_{k-1}^{\mathsf{T}}$, it follows that $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}})$ is a numeral, which implies that $\mathit{eval}(y, \tau(a_{y,d}))$ is defined after all. We may therefore set $t = \mathit{evaltime}(y, \tau(a_{y,d}))$ and $u = u(t)$, so that $u \in H$; clearly u is computable from y and d. We will show that u signifies a KLS modulus of $\dot{G}$ at d, p as required.

Suppose from here on that $\dot{G}$ is extensional and represents $G \in \mathsf{HEO}_{k-1}$. We first claim that $\mathit{eval}(y, \tau(a_{y,d})) = p$. Suppose not; then as before we have $a_{y,d} \bullet i = d \bullet i$ for all i, so that $\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}$ represents h. But now since $\dot{G}$ is extensional we have $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}) = \dot{G} \cdot \dot{h} = \widehat{p}$, implying that $\mathit{eval}(y, \tau(a_{y,d})) = p$ after all.

Let $\mathfrak{b} = \bigsqcup_{i < time(u)} \zeta_{k-2}(d \bullet i)$, and for any $c \in \mathbb{N}$, let $x_c \in \mathsf{HEO}_{k-2}$ be the element represented by the element $\mathfrak{x}_c \in \mathsf{Tot}(\mathsf{P}_{k-2}^{\mathit{eff}})$ from Theorem 4.9. We claim that $G(x_c) = p$ whenever $\zeta_{k-2}(c) \sqsupseteq \mathfrak{b}$. First, if $\zeta_{k-2}(c) \sqsupseteq \mathfrak{b}$ then $c = \mathrm{filter}(b(d, t), c)$, so $\eta_{d,u,c}$ basically enumerates $\mathfrak{x}_c$; but $\mathit{bullet} \cdot (\delta(d, u, c))\widehat{\ }$ represents $\eta_{d,u,c}$, so $\mathit{std\text{-}dense} \cdot \widehat{\langle d, u \rangle} \cdot \widehat{c} = \nabla_{k-2} \cdot (\mathit{bullet} \cdot (\delta(d, u, c))\widehat{\ })$ represents x_c by the main induction hypothesis. So if $G(x_c) \neq p$ then $\dot{G} \cdot (\mathit{std\text{-}dense} \cdot \widehat{\langle d, u \rangle} \cdot \widehat{c}) \neq \widehat{p}$. Let m be the smallest number such that $\dot{G} \cdot (\mathit{std\text{-}dense} \cdot \widehat{\langle d, u \rangle} \cdot \widehat{m})$ (which is always a numeral) is not $\widehat{p}$. Clearly $\mathit{easy\text{-}critical\text{-}index} \cdot \dot{G} \cdot \widehat{\langle d, u \rangle} \cdot \widehat{p} = \widehat{m}$, whence $\iota(y, d, u, p) = m$. It follows that $a_{y,d} \bullet i = \eta_{d,u,m}(i)$ for all i; hence $\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}$ and $\mathit{std\text{-}dense} \cdot \widehat{\langle d, u \rangle} \cdot \widehat{m}$ represent the same element of HEO_{k-2}. But now since $\dot{G}$ is extensional we have $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}) = \dot{G} \cdot (\mathit{std\text{-}dense} \cdot \widehat{\langle d, u \rangle} \cdot \widehat{m}) \neq \widehat{p}$, which contradicts the fact that $\mathit{eval}(y, \tau(a_{y,d})) = p$.

The remainder of the proof is similar to the end of the proof of Theorem 8.4. Suppose

$\dot{h}' \in \mathsf{Ext}(A_{k-2})$ represents some $h' \in U_{\mathfrak{b}}$ such that $\dot{G} \cdot \dot{h}' = \widehat{p'} \neq \widehat{p}$. By applying the above argument to some d' indexing a basic enumeration of $\dot{h}'$, we may obtain an element $\mathfrak{b}'$ with $h' \in U_{\mathfrak{b}'}$ such that $G(x_c) = p'$ whenever $\zeta_{k-2}(c) \sqsupseteq \mathfrak{b}'$. Since $h' \in U_{\mathfrak{b}} \cap U_{\mathfrak{b}'}$, by Proposition 4.10(i) we have that $\mathfrak{b}, \mathfrak{b}'$ are consistent. So if c is a code for $\mathfrak{b} \sqcup \mathfrak{b}'$ then $p = G(x_c) = p'$, which is a contradiction. □

Using this, we may obtain a correspondingly strengthened version of Proposition 8.5:

Proposition 9.10. Suppose given $e \in \mathbb{N}$, $\dot{\lambda} \in \mathsf{Ext}(A_1)$, $\dot{G} \in A^{\mathsf{T}}_{k-1}$, and $d, \mathfrak{h}, h, \dot{h}, p$ as in Proposition 9.9. Then there exists $u' \in H$ such that

(1) $\mathit{satisfies} \cdot \dot{G} \cdot \dot{\lambda} \cdot \widehat{e} \cdot \widehat{p} \cdot \widehat{d} \cdot \widehat{u'} = tt$
(2) u' signifies a modified KLS modulus of $\dot{G}$ at d, p in the sense of the previous proposition.

Moreover, a suitable value for u' may be computed uniformly from e, d, any realizer $w \vdash^{\gamma}_1 \lambda$, and any realizer $y \vdash^{\gamma}_{k-1} \dot{G}$.

Proof. Precisely analogous to the proof of Proposition 8.5. Note that this yields a value for u' such that $\mathit{time}(u') \geq t$, where t is the KLS modulus $\mathit{evaltime}(y, \tau(a_{y,d}))$ resulting from the proof of Proposition 9.9. (This fact will be used in the proof of Proposition 9.13.) □

We will write $u(e, d, w, y)$ for the computed value of u' given by the proof of the above proposition, and say that a basic signifier $\langle\langle d, u', p\rangle\rangle$ is *satisfactory* for $\dot{G} \in A^{\mathsf{T}}_{k-1}$ (relative to $\dot{\lambda}$ and e) if conditions (1) and (2) of the above proposition are satisfied.

A version of Lemma 8.6 goes through under the weaker assumption that $\dot{G} \in A^{\mathsf{T}}_{k-1}$.

Lemma 9.11. Suppose $e \in \mathbb{N}$, $\dot{\lambda} \in \mathsf{Ext}(A_1)$, $\dot{G} \in A^{\mathsf{T}}_{k-1}$, $w \vdash^{\gamma}_1 \dot{\lambda}$ and $y \vdash^{\gamma}_{k-1} \dot{G}$. Then for any $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$ there is some index $d \in D$ such that $\mathfrak{b}(e, d, w, y) \sqsubseteq \mathfrak{h}$.

Proof. As for Lemma 8.6. Note that the set D here is intuitively "larger" than the set D used in Section 8. □

Our set D would certainly suffice for obtaining an analogue of Proposition 8.7, but later parts of the proof require that we work with a somewhat richer set. Using notation from the proof of Proposition 9.9, let us say a pair (d, y) is *acceptable* if either $\mathit{eval}(y, \tau(a_{y,d})) \neq \mathit{eval}(y, \tau(d))$ (both sides being defined), or both $\mathit{eval}(y, \tau(a_{y,d})) = \mathit{eval}(y, \tau(d))$ and $\iota(y, d, u(t), p)\downarrow$ where $t = \mathit{evaltime}(y, \tau(a_{y,d}))$ and $p = \mathit{eval}(y, \tau(d))$. Now let E denote the closure of D under the following rule: if $d \in E$, $y \in \mathbb{N}$ and (d, y) is acceptable, then $a_{y,d} \in E$.

It is easy to see by induction that every $d \in E$ is an index for a basic enumeration of some $\mathfrak{x}_c$. More specifically, any element of E may be described syntactically by a tuple $\Delta = (d, v, m, d_0, y_0, \ldots, d_{r-1}, y_{r-1}, d_r)$ $(r \geq 0)$ with the following properties: $(d, v, m) \in M$, $d_0 = \delta(d, v, m)$, and for each $i < r$, (d_i, y_i) is acceptable and $d_{i+1} = a_{y_i, d_i}$. Such a tuple may be regarded as a proof that $d_r \in E$. Let L denote the set of all such tuples Δ, and for $\Delta \in L$, let $\epsilon(\Delta)$ denote simply the last component of Δ. Clearly both L and E are c.e., and ϵ effectively maps L onto E. Furthermore, we may define a computable function

$\kappa : L \to \mathbb{N}$ such that $\kappa(\Delta)$ is some ζ_{k-2}-code c where $\epsilon(\Delta)$ indexes a basic enumeration of $\mathfrak{x}_c$. The definition of $\kappa(\Delta)$ is by induction on the length of Δ:

— $\kappa((d, v, m, d_0)) = \text{filter}\,(b(d, \mathit{time}(v)), m)$;
— if $\kappa((d, v, m, d_0, y_0, \ldots, d_r)) = c$ and $\mathit{eval}(y_r, \tau(a_{y_r,d_r})) \neq \mathit{eval}(y_r, \tau(d_r))$
then $\kappa((d, v, m, d_0, y_0, \ldots, d_r, y_r, d_{r+1})) = c$;
— if $\kappa((d, v, m, d_0, y_0, \ldots, d_r)) = c$, $\mathit{eval}(y_r, \tau(a_{y_r,d_r})) = \mathit{eval}(y_r, \tau(d_r)) = p$,
$\mathit{evaltime}(y_r, \tau(a_{y_r,d_r})) = t$ and $\iota(y_r, d_r, u(t), p) = m$,
then $\kappa((d, v, m, d_0, y_0, \ldots, d_r, y_r, a_{y_r,d_r})) = \text{filter}\,(b(d_r, t), m)$.

We now give an analogue of Proposition 8.7 as follows:

Proposition 9.12. Given $e \in \mathbb{N}$, $\dot\lambda \in \mathsf{Ext}(A_1)$ and $\dot G \in A^{\mathsf{T}}_{k-1}$, there is a sequence $c_0, c_1, \ldots$ of basic type $k-1$ codes such that

(1) each $\zeta_{k-1}(c_j)$ has some basic signifier $\langle\langle d_j, u_j, p_j\rangle\rangle$ that is satisfactory for $\dot G$ relative to $\dot\lambda$ and e;
(2) if $\dot G \in \mathsf{Ext}(A_{k-1})$, then the $\zeta_{k-1}(c_j)$ are pairwise consistent;
(3) if the $\zeta_{k-1}(c_j)$ are pairwise consistent, then the element $\mathfrak{G} = \bigsqcup_j \zeta_{k-1}(c_j)$ is in $\mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-1})$ and so represents some $G \in \mathsf{HEO}_{k-1}$.

Moreover, the codes c_j, and suitable associated signifiers $\langle\langle d_j, u_j, p_j\rangle\rangle$, are computable from j uniformly in $e, w \vdash^{\gamma}_1 \dot\lambda$ and $y \vdash^{\gamma}_{k-1} \dot G$.

Proof. Analogous to the proof of Proposition 8.7. We use the c.e. set E rather than D for reasons that will appear in the proof of Proposition 9.13 below. We construct the signifiers $\langle d_j, u_j, p_j\rangle$ from a computable enumeration $d_0, d_1, \ldots$ of E by taking $u_j = u(e, d_j, w, y)$ (using the new definition of the function u) and $\widehat{p_j} = \dot G{\cdot}(\mathit{total\text{-}elt}{\cdot}\widehat{d_j})$. Property (1) and the uniform computability are then immediate, and the proof of pairwise consistency used in Proposition 8.7 goes through under the assumption that $\dot G \in \mathsf{Ext}(A_{k-1})$. For property (3), it suffices to show that for any $\mathfrak{h} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_{k-2})$ there is some $c_j = b_j \mapsto \breve{p}_j$ with $\zeta_{k-2}(b_j) \sqsubseteq \mathfrak{h}$. But this is immediate from Lemma 9.11 and the fact that $D \subseteq E$. □

It will be useful to characterize those $\dot G$ for which the elements $\zeta_{k-1}(c_j)$ generated as above are pairwise consistent. Let us say $\dot G$ is *quasi-extensional* if for any $\Delta, \Delta' \in L$ such that $\kappa(\Delta) = \kappa(\Delta')$ (on the nose!), setting $d = \epsilon(\Delta)$ and $d' = \epsilon(\Delta')$ we have $\dot G{\cdot}(\mathit{total\text{-}elt}{\cdot}\widehat{d}) = \dot G{\cdot}(\mathit{total\text{-}elt}{\cdot}\widehat{d'})$. We also say $\dot G$ *quasi-represents* a functional $G \in \mathsf{HEO}_{k-1}$ if whenever $d \in E$ and $\dot h = \mathit{total\text{-}elt} \cdot \widehat{d}$ represents $h \in \mathsf{HEO}_{k-2}$, we have $\dot G \cdot \dot h = \widehat{G(h)}$. Clearly, if $\dot G$ represents G then $\dot G$ quasi-represents G. Furthermore, if $\dot G$ quasi-represents G then $\dot G$ is quasi-extensional, since for any $\Delta \in L$ we have that $d = \epsilon(\Delta) \in E$ indexes a basic enumeration of $\mathfrak{x}_{\kappa(\Delta)}$, so that $\mathit{total\text{-}elt} \cdot \widehat{d}$ represents the corresponding element $x_{\kappa(\Delta)} \in \mathsf{HEO}_{k-2}$ and $\dot G \cdot (\mathit{total\text{-}elt} \cdot \widehat{d}) = (G(x_{\kappa(\Delta)}))\widehat{\ }$.

An important point here is that if $\dot G$ is not quasi-extensional, then since E is c.e. we will be able to find a counterexample by means of an effective search which may be performed within A itself (see Proposition 9.17 below). On the other hand, the following proposition (corresponding roughly to Proposition 9.4) shows that the quasi-extensional elements are as well-behaved as the extensional ones for our purposes.

Proposition 9.13. Suppose given $\dot{G} \in A^{\mathsf{T}}_{k-1}$ and $e, w, y \in \mathbb{N}$ as in the above proposition. Then the compact elements $\zeta_{k-1}(c_j)$ generated from e, w, y as above are pairwise consistent iff $\dot{G}$ is quasi-extensional. Moreover, in this case, $\dot{G}$ quasi-represents some unique $G \in \mathsf{HEO}_{k-1}$, which is also the functional represented by $\mathfrak{G} = \bigsqcup_j \zeta_{k-1}(c_j)$.

Proof. First suppose the $\zeta_{k-1}(c_j)$ are pairwise consistent. By Proposition 9.12, the element $\mathfrak{G} = \bigsqcup \zeta_{k-1}(c_j)$ is total and represents some $G \in \mathsf{HEO}_{k-1}$; we will show that $\dot{G}$ quasi-represents G. Given any element $\dot{h} = \mathit{total\text{-}elt} \cdot \widehat{d}$ where $d \in E$, suppose d arises as d_j in our enumeration of E, and consider the corresponding code c_j and signifier $\langle d_j, u_j, p_j \rangle$ generated using e, w, y. From the proof of Proposition 9.12, we see that p_j was chosen so that $\widehat{p_j} = \dot{G} \cdot \dot{h}$, and $\zeta_{k-1}(c_j) = \mathfrak{b}_j \Rightarrow p_j$ where $\mathfrak{b}_j = \bigsqcup_{i < time(u_j)} \zeta_{k-2}(d_j \bullet i)$. Now since $\zeta_{k-1}(c_j) \sqsubseteq \mathfrak{G}$ and $\mathfrak{G}$ represents G, we have that $G(h') = p_j$ for all $h' \in U_{\mathfrak{b}_j}$. But the element h represented by $\dot{h}$ belongs to $U_{\mathfrak{b}_j}$ since it is represented in $\mathsf{P}^{\mathit{eff}}_{k-2}$ by $\bigsqcup_i \zeta_{k-2}(d_j \bullet i) \sqsupseteq \mathfrak{b}_j$, so $G(h) = p_j$. Thus $\dot{G}$ quasi-represents G, so $\dot{G}$ is quasi-extensional.

For the uniqueness, suppose $\dot{G}$ quasi-represents both G and G', and consider an arbitrary $h \in \mathsf{HEO}_{k-2}$. Let $U_{\mathfrak{b}}, U_{\mathfrak{b}'}$ be neighbourhoods for the continuity of G, G' respectively at h; then $\mathfrak{b}, \mathfrak{b}'$ are consistent and we may find $d \in E$ so that $\mathit{total\text{-}elt} \cdot \widehat{d}$ represents an element of $U_{\mathfrak{b} \sqcup \mathfrak{b}'}$. But now $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d})$ represents both $G(h)$ and $G'(h)$, so $G(h) = G'(h)$.

For the remaining implication, suppose $\dot{G}$ is quasi-extensional, and that $c_j = b_j \mapsto \breve{p_j}$ and $c_{j'} = b_{j'} \mapsto \breve{p_{j'}}$ are among the codes generated from e, w, y. Suppose too that $\mathfrak{b}_j = \zeta_{k-2}(b_j)$ and $\mathfrak{b}_{j'} = \zeta_{k-2}(b_{j'})$ are consistent, otherwise $\zeta_{k-1}(c_j)$ and $\zeta_{k-1}(c_{j'})$ are trivially consistent. Let $\langle d_j, u_j \rangle$ and $\langle d_{j'}, u_{j'} \rangle$ be the pre-signifiers for $\mathfrak{b}_j, \mathfrak{b}_{j'}$ generated as in Proposition 9.12, and let m be a code for $\mathfrak{b}_j \sqcup \mathfrak{b}_{j'}$. Then $(d_j, u_j, m), (d_{j'}, u_{j'}, m) \in M$, and if $d^*_j = \delta(d_j, u_j, m)$ and $d^*_{j'} = \delta(d_{j'}, u_{j'}, m)$ then $\kappa((d_j, u_j, m, d^*_j)) = \kappa((d_{j'}, u_{j'}, m, d^*_{j'})) = m$, so by quasi-extensionality we have $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^*_j}) = \dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^*_{j'}})$. We will show that $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^*_j}) = \widehat{p_j}$; by symmetry we will then also have $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^*_{j'}}) = \widehat{p_{j'}}$, whence $p_j = p_{j'}$ and the elements $\zeta_{k-1}(c_j)$, $\zeta_{k-1}(c_{j'})$ are consistent.

We henceforth write $d, d^*, u, p, \mathfrak{b}$ for $d_j, d^*_j, u_j, p_j, \mathfrak{b}_j$ respectively. Recall from the foregoing proofs that $d \in E$, that p was chosen so that $\widehat{p} = \dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d})$, and that u was chosen so that $\mathit{time}(u) \geq \mathit{time}(u^-)$, where $u^- = u(\mathit{evaltime}(y, \tau(a_{y,d})))$, borrowing notation from the proof of Proposition 9.9. Let $z^- = \langle d, u^- \rangle$ and let $\mathfrak{b}^-$ be the element pre-signified by z^-; note that $\mathfrak{b}^- \sqsubseteq \mathfrak{b}$. Note also that $(d, u^-, m) \in M$, and setting $d^{*-} = \delta(d, u^-, m)$ we again have $\kappa(d, u^-, m, d^{*-}) = m$, so again by quasi-extensionality we have $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^*}) = \dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^{*-}})$.

Now suppose for contradiction that $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^{*-}}) = \widehat{p'} \neq \widehat{p}$. By examining the code in the proof of Proposition 8.4, we will show that this "counterexample" (or a similar one) will be detected by the program `easy-critical-index`, causing $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}) \neq \widehat{p}$ and contradicting the choice of u^-.

First, from the definition of `std-dense` in the proof of Proposition 9.9, we see that $\dot{G} \cdot (\mathit{std\text{-}dense} \cdot \widehat{z^-} \cdot \widehat{m}) = \dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d^{*-}}) = \widehat{p'}$, so that $\mathit{is\text{-}counterex\text{-}for} \cdot \dot{G} \cdot \widehat{z^-} \cdot \widehat{p} \cdot \widehat{m} = \mathit{tt}$. Since $\dot{G} \cdot (\mathit{std\text{-}dense} \cdot \widehat{z^-} \cdot \widehat{m'})$ is a numeral for every m', it follows that $\mathit{easy\text{-}critical\text{-}index} \cdot \dot{G} \cdot \widehat{z^-} \cdot \widehat{p}$ yields some $\widehat{m_0}$ with $\dot{G} \cdot (\mathit{std\text{-}dense} \cdot \widehat{z^-} \cdot \widehat{m_0}) \neq \widehat{p}$.

Now consider the index $a_{y,d}$ as in the proof of Proposition 9.9. Recall that

$$a_{y,d} \bullet i \;\simeq\; \begin{cases} d \bullet i & \text{if } i < \mathit{evaltime}(y, \tau(a_{y,d})) \\ d \bullet i & \text{if } i \geq \mathit{evaltime}(y, \tau(a_{y,d})) \text{ and } \mathit{eval}(y, \tau(a_{y,d})) \neq p \\ \xi_{k-2,c}(i) & \text{if } i \geq \mathit{evaltime}(y, \tau(a_{y,d})) = t,\ \mathit{eval}(y, \tau(a_{y,d})) = p, \\ & \quad m_0 = \iota(y, d, u(t), p) \text{ and } c = \mathrm{filter}(b(d,t), m_0) \end{cases}$$

and that $\mathit{eval}(y, \tau(d)) = p$ and $\mathit{eval}(y, \tau(a_{y,d}))$ is defined. We may now re-run parts of the proof of Proposition 9.9 using the fact that $\dot{G}$ is quasi-extensional. We first claim that $\mathit{eval}(y, \tau(a_{y,d})) = p$. Suppose not; then the pair (d, y) is acceptable and $d \in E$, so $a_{y,d} \in E$. Indeed, if d arises as $\epsilon(\Delta)$ for $\Delta \in L$, then $a_{y,d}$ arises as $\epsilon(\Delta')$ where Δ' is obtained by appending y, d to Δ, and moreover $\kappa(\Delta') = \kappa(\Delta)$. So by quasi-extensionality we have $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}) = \dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{d}) = \widehat{p}$, whence $\mathit{eval}(y, \tau(a_{y,d})) = p$ after all.

We now have that $\mathit{eval}(y, \tau(a_{y,d})) = \mathit{eval}(y, \tau(d)) = p$, and also that $\mathit{easy\text{-}critical\text{-}index} \cdot \dot{G} \cdot \langle d, u^- \rangle\widehat{\ } \cdot \widehat{p} = \widehat{m_0}$ where $u^- = u(t)$ for $t = \mathit{evaltime}(y, \tau(a_{y,d}))$, so that $\iota(y, d, u(t), p) = m_0$. Thus (d, y) is an acceptable pair and $d \in E$, so $a_{y,d} \in E$. Moreover, if $\Delta' \in L$ represents the generation of $a_{y,d}$ via d, then we have

$$\kappa(\Delta') \;=\; \mathrm{filter}\,(b(d,t), m_0) \;=\; \kappa((d, u^-, m_0, \delta(d, u^-, m_0)))$$

So by quasi-extensionality again we have

$$\begin{aligned} \dot{G} \cdot (\mathit{total\text{-}elt} \cdot \widehat{a_{y,d}}) \;&=\; \dot{G} \cdot (\mathit{total\text{-}elt} \cdot (\delta(d, u^-, m_0))\widehat{\ }) \\ &=\; \dot{G} \cdot (\mathit{std\text{-}dense} \cdot \widehat{z^-} \cdot \widehat{m_0}) \;\neq\; \widehat{p} \end{aligned}$$

which contradicts the fact that $\mathit{eval}(y, \tau(a_{y,d})) = p$. □

Continuing with our adaptation of the argument of Section 8, we have the following variant of Corollary 8.8. Here, as before, we suppose that $\nu \in K_2^{\mathit{eff}}$ is a basic enumeration of some $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}_k^{\mathit{eff}})$ representing $\Phi \in \mathsf{HEO}_k$.

Corollary 9.14. For any $e \in \mathbb{N}$, $\dot{\lambda} \in \mathsf{Ext}(A_1)$, $w \vdash_1^\gamma \dot{\lambda}$, $\dot{G} \in A_{k-1}^{\mathsf{T}}$ and $y \vdash_{k-1}^\gamma \dot{G}$, there exist a position index j for ν and a number

$$s \;=\; \langle\langle d_0, u_0, p_0\rangle, \ldots, \langle d_{r-1}, u_{r-1}, p_{r-1}\rangle\rangle$$

such that

(1) for each $i < r$, $\langle\langle d_i, u_i, p_i\rangle\rangle$ is satisfactory for $\dot{G}$ relative to $\dot{\lambda}$ and e;
(2) *either*
  (2.1) $\nu(j)$ is a basic code $\langle c \mapsto \breve{q}\rangle$, s is a well-formed type $k-1$ signifier, and $\zeta_{k-1}(c) = [\![\, s \,]\!]$, *or*
  (2.2) $r = 2$ and the basic signifiers $\langle\langle d_0, u_0, p_0\rangle\rangle, \langle\langle d_1, u_1, p_1\rangle\rangle$ denote inconsistent elements of $\mathsf{P}_{k-1}^{\mathit{comp}}$ (there is no condition on j in this latter case);
(3) if $\dot{G}$ quasi-represents G, then we are in case (2.1) with $q = \Phi(G)$.

Moreover, suitable values of j and s are computable from $\nu \in \mathbb{N}^{\mathbb{N}}$ and $e, w, y \in \mathbb{N}$ by means of a type 2 partial computable functional.

Proof. If $\dot{G}$ quasi-represents G, the search described in the proof of Corollary 8.8 (using e, w, y) will yield a pair (j, s) satisfying conditions (1) and (2.1), and in fact we will have $q = \Phi(G)$. We may therefore interleave this search with a search for a pair (i, i') such that the generated elements $\zeta_{k-2}(c_i), \zeta_{k-2}(c_{i'})$ are inconsistent. If these are found, we may set j arbitrarily and take $s = \langle\langle d_i, u_i, p_i\rangle, \langle d_{i'}, u_{i'}, p_{i'}\rangle\rangle$, so that conditions (1) and (2.2) are satisfied. By Proposition 9.13, for any $\dot{G} \in A^{\mathsf{T}}_{k-1}$ at least one of these searches will succeed, and if the second succeeds then $\dot{G}$ is not quasi-extensional. □

From this point on, the argument proceeds much as in the modified continuous case. We modify our definition of satisfactory representation to include all possible "inconsistent signifiers", in order to catch all the non-extensional elements $\dot{G}$.

Definition 9.15. A *modified satisfactory representation* of $\Phi \in \mathsf{HEO}_k$ is a total computable function $\lambda : \mathbb{N} \to \mathbb{N}$, where $\lambda(j) = \langle e_j, s_j, q_j\rangle$ for each j, such that

(1) any number $e \in \mathbb{N}$ appears as e_j for at most one j;
(2) for each j, *either*
  (2.1) s_j is a signifier $\langle\langle d_{j0}, u_{j0}, p_{j0}\rangle, \ldots, \langle d_{j(r_j-1)}, u_{j(r_j-1)}, p_{j(r_j-1)}\rangle\rangle$ for some compact element $\mathfrak{c}_j \in \mathsf{P}_{k-1}$, and $\mathfrak{c}_j \Rightarrow q_j$ is consistent with Φ, *or*
  (2.2) s_j is of the form $\langle\langle d_{j0}, u_{j0}, p_{j0}\rangle, \langle d_{j1}, u_{j1}, p_{j1}\rangle\rangle$, and the signifiers $\langle\langle d_{j0}, u_{j0}, p_{j0}\rangle\rangle$, $\langle\langle d_{j1}, u_{j1}, p_{j1}\rangle\rangle$ denote inconsistent elements of P^{comp}_{k-1}, and $q_j = 0$;
(3) for any $\dot{\lambda}' \in \mathsf{Ext}(A_1)$ and $\dot{G} \in A^{\mathsf{T}}_{k-1}$, there exists a position index j for λ such that each $\langle\langle d_{ji}, u_{ji}, p_{ji}\rangle\rangle$ associated with s_j is satisfactory for $\dot{G}$ relative to $\dot{\lambda}'$ and e_j.

Proposition 9.16. Suppose $\nu \in K_2^{\mathit{eff}}$ basically enumerates some $\mathfrak{Phi}$ representing $\Phi \in \mathsf{HEO}_k$. Then a modified satisfactory representation λ^ν of Φ exists and is computable from ν by means of a type 2 partial computable functional.

Proof. Similar to the proof of Proposition 8.10. Let $\theta : \mathbb{N}^{\mathbb{N}} \times \mathbb{N}^3 \rightharpoonup \mathbb{N}^2$ be the uniformly partial computable function embodying the composite search for (j, s) in the proof of Corollary 9.14, so that $\theta(\nu, e, y, w) = (j, s)$. Having found j and s, a suitable value for q is given by

$$q(j,s) \;=\; \begin{cases} q & \text{if } s \text{ is a consistent signifier and } \nu(j) = c \mapsto \breve{q} \text{ for some } c \\ 0 & \text{otherwise} \end{cases}$$

Now consider the set

$$M^\nu \;=\; \{\langle e, s, q\rangle \mid \exists y, w, j.\;\; e = \langle y, w\rangle \wedge \theta(\nu, e, y, w) = (j, s) \wedge q = q(j, s)\}$$

Then as before, M^ν is c.e. uniformly in ν, and the rest of the proof proceeds as for Proposition 8.10. An inspection of the search procedure confirms that even if y, w do not realize suitable elements $\dot{G}, \dot{\lambda}'$, the value of $\theta(\nu, \langle y, w\rangle, y, w)$, if any, will still satisfy condition (2.1) or (2.2) of Definition 9.15 as appropriate. □

As before, we may therefore construct an NRComb program `satis-rep` : $\overline{1} \to \overline{1}$ such that whenever $\dot{\nu} \in \mathsf{Ext}(A_1)$ represents a basic enumeration ν of some $\mathfrak{Phi} \in \mathsf{Tot}(\mathsf{P}^{\mathit{eff}}_k)$, the element *satis-rep* $\cdot\, \dot{\nu} \in A_1$ represents the modified satisfactory representation λ^ν.

Next, we introduce some machinery to allow us to perform a search within A for a counterexample to the quasi-extensionality of $\dot{G}$. If $\Delta = (d, v, m, d_0, y_0, \ldots, d_r) \in L$, we write $\langle\Delta\rangle$ for $\langle d, v, m, d_0, y_0, \ldots, d_r\rangle$. Suppose $\texttt{L-enum} : \overline{0} \to \overline{0}$ represents a computable enumeration of the set of all $\langle\Delta\rangle$ where $\Delta \in L$, $\texttt{eps} : \overline{0} \to \overline{0}$ is such that $\mathit{eps} \cdot \widehat{\langle\Delta\rangle} = (\epsilon(\Delta))\widehat{\ }$ for all $\Delta \in L$, and $\texttt{kap} : \overline{0} \to \overline{0}$ is such that $\mathit{kap} \cdot \widehat{\langle\Delta\rangle} = (\kappa(\Delta))\widehat{\ }$ for all $\Delta \in L$. We now define:

$$\begin{aligned}
\texttt{incons2 G}^{k-1}\ \texttt{Delta0}^0\ \texttt{Delta1}^0 \quad &\equiv \quad \texttt{and (eq (kap Delta0) (kap Delta1))} \\
&\qquad\qquad \texttt{(not (eq (G (total-elt (eps Delta0)))} \\
&\qquad\qquad\qquad \texttt{(G (total-elt (eps Delta1)))))} \\
\texttt{incons G}^{k-1}\ \texttt{n}^0 \quad &\equiv \quad \texttt{incons2 G (L-enum (proj0 n))} \\
&\qquad\qquad\qquad \texttt{(L-enum (proj1 n))}
\end{aligned}$$

The key property of `incons` is as follows:

Proposition 9.17. For any $\dot{G} \in A_{k-1}^{\mathsf{T}}$ and $n \in \mathbb{N}$ we have $\mathit{incons} \cdot \dot{G} \cdot \widehat{n} \in \{\mathit{tt}, \mathit{ff}\}$. Moreover, $\dot{G}$ is quasi-extensional iff $\mathit{incons} \cdot \dot{G} \cdot \widehat{n} = \mathit{ff}$ for all n.

Proof. The first statement follows from the fact that for any n, $\mathit{eps} \cdot (\mathit{L\text{-}enum} \cdot \widehat{n})$ represents an element of E, so that $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot (\mathit{eps} \cdot (\mathit{L\text{-}enum} \cdot \widehat{n})))$ evaluates to a numeral, and moreover `proj0`, `proj1` represent total functions. Moreover, the definition of `incons2` is a direct translation of the condition that $\Delta_0, \Delta_1 \in L$ constitute a counterexample to the quasi-extensionality of $\dot{G}$, namely, that $\kappa(\Delta_0) = \kappa(\Delta_1)$ but $\dot{G} \cdot (\mathit{total\text{-}elt} \cdot (\epsilon(\Delta_0))\widehat{\ }) \neq \dot{G} \cdot (\mathit{total\text{-}elt} \cdot (\epsilon(\Delta_1))\widehat{\ })$. The second statement then follows easily. □

We now define the Normann program $\texttt{nabla}_k$ just as in Section 8.4, except that we modify the definition of `test-pre-critical` as follows:

$$\begin{aligned}
\texttt{refutes-or-incons G}^{k-1}\ \texttt{x}^0\ \texttt{m}^0 \quad &\equiv \quad \texttt{or (refutes G x m) (incons G m)} \\
\texttt{test-pre-critical G}^{k-1}\ \texttt{lam}^1\ \texttt{x}^0\ \texttt{m}^0 \quad &\equiv \quad \texttt{if}_0\ \texttt{(or (refutes G x m)} \\
&\qquad\qquad\qquad\qquad \texttt{(eq (proj2 x) (proj2 (lam m))))} \\
&\qquad\qquad\ \texttt{m}\ \ \texttt{(min (refutes-or-incons G x))}
\end{aligned}$$

It remains to check that the element ∇_k satisfies the properties required by Lemma 9.8:

Proposition 9.18. Suppose $\Phi \in \mathsf{HEO}_k$, $\nu \Vdash_k^\delta \Phi$, $\dot{\nu} \in \mathsf{Ext}(A_1)$ represents ν and $\dot{G} \in A_{k-1}^{\mathsf{T}}$. Then

(1) $\nabla_k \cdot \dot{\nu} \cdot \dot{G}$ evaluates to a numeral;
(2) if $\dot{G}$ quasi-represents $G \in \mathsf{HEO}_{k-1}$, then $\nabla_k \cdot \dot{\nu} \cdot \dot{G} = \widehat{\Phi(G)}$.

The proof is a mild elaboration of the proof of Theorem 8.13. As in the modified continuous setting, we argue by cases according to whether or not $\dot{G}$ is quasi-extensional; the remarks at the end of Section 9.1 apply here *mutatis mutandis*.

10. Commentary on the main theorems

In this section we collect together some technical remarks and observations concerning our theorems, including some mild generalizations and extensions. We start by surveying some examples and non-examples of NR-TPCAs to which our theorems can be applied, in order to give an indication of the scope and limitations of our results.

10.1. *Scope and applications of our theorems*

As the range of examples of NR-TPCAs in Section 3 makes clear, a large number of new characterizations of the type structures C, $\mathsf{C}^{\mathit{eff}}$, HEO flow from our theorems. Several of these NR-TPCAs have their own intrinsic appeal, and the corresponding characterizations can thus be seen as having some interest in their own right.

10.1.1. *Typed models.* Among full continuous NR-TPCAs that satisfy the conditions Theorem 5.13 are many of the models of an "algebraic" character that are studied in denotational semantics, such as (the finite type portions of) the Milner model of PCF, Berry's *stable* model, the Bucciarelli-Ehrhard *strongly stable* model, the Berry-Curien *sequential algorithms* model, and most of the *game models* considered by Abramsky, Hyland *et al*, in both their intensional and extensional manifestations. In all these models, condition (C) of Theorem 5.13 typically holds because we may take the canonical representative of a function $\mathbb{N} \to \mathbb{N}$ to be the strict map $N_\perp \to N_\perp$ which represents it, and there is a strictifying map $N_\perp^{N_\perp} \to N_\perp^{N_\perp}$ which plays the role of **norm**. In addition, it is worth remarking that our theorems apply equally well to the call-by-name, call-by-value and lazy variants of the above models.

The kinds of full continuous models for which condition (C) fails are highly intensional models, often with a syntactic flavour, such as the term model NRC^∞ itself. (Our theorem does of course apply to $\mathsf{NRC}^{\infty+}$.) Perhaps a more serious shortcoming is that condition (AB) tends to fail in models in which the numerals are not maximal elements, such as $\mathcal{P}\omega$ or the model L given by continuous lattices. We will discuss the prospects for weakening some of these restrictions in Section 10.4.

For all of the positive examples listed above, one can identify an "effective submodel", and in some cases (such as the Milner and strongly stable models), more than one candidate for such a structure can be identified. In all these cases, Theorem 5.14 shows that the corresponding relative extensional collapse is $\mathsf{C}^{\mathit{eff}}$, and Theorem 5.12 shows that the extensional collapse of the effective model considered in its own right is HEO.

Also of interest are the term models for various simply-typed programming languages, such as the models NRC and PCF from Section 3.4. A sample of other models is considered in (Longley 1999a). In many good cases, these models coincide with the effective submodels of one of the denotational models listed above, but in other cases no good denotational counterpart is known. It is intuitively clear that conditions (A) and (B) of Theorem 5.12 will be satisfied by the term model for any reasonable programming language with an effective, deterministic operational semantics. This includes even highly

"intensional" languages such as PCF + `timeout` or PCF + `quote` (in these cases, the corresponding infinitary term models fail to satisfy condition (C)).

It is worth pointing out that our results apply even to such mathematically unwieldy structures as term models for full-scale programming languages. Consider, for example, the set of "closed" programs of simple type in Standard ML (Milner *et al.* 1997). (The notion of closed program that we have in mind requires some elaboration, and we will not give details here. The idea is to permit *local* uses of exceptions, state variables and other non-functional features, but to disallow the manipulation of *global* state variables.) Under the congruence generated by the evaluation relation, these clearly form an effective NR-TPCA, and so without knowing any more about the definition of ML, we may conclude that the hereditarily total functionals computable in ML are precisely those of HEO.

An interesting test case is provided by term models for *non-deterministic* languages. Here it is natural to take the numeral $\widehat{n}$ to correspond to the set of programs that *must* evaluate to n under any possible sequence of non-deterministic choices. In the case of a language with a binary non-deterministic choice operator, we see by König's Lemma that the set of such programs is c.e., and so Theorem 5.12 applies. However, this is not the case for a language with countable non-deterministic choice, and such languages fall outside the scope of our theorem.

10.1.2. *Untyped models.* Many untyped PCAs provide pleasing examples of our theorems. However, some of these, such as Plotkin's $\mathbb{T}^\omega$ (Plotkin 1978) and van Oosten's $\mathcal{B}$ (van Oosten 1999), are known to be "equivalent" to one of the typed models already mentioned, and so should not really be counted as distinct instances — see (Longley 1999a). Perhaps the most striking applications in the untyped world are to syntactic models, such as term models for untyped combinatory logic and various untyped lambda calculi. The problem of identifying the total type structure arising from the usual untyped lambda calculus has occasionally been aired as an open question (see *e.g.* (Beeson 1985, Section IV.8) or (Longley 1995, page 250)). Prior to obtaining the theorems in this paper, the author believed that this problem was probably intractable at present. (This opinion was based largely on his experience of trying to identify the *partial* type structure arising from this model — see (Longley 1995, Section 7.4).) However, it now follows immediately from Theorem 5.12 that $\mathsf{EC}(\Lambda^0/T) \cong \mathsf{HEO}$, where Λ^0 is the set of closed terms of the untyped lambda calculus, and T is any sub-theory of the theory induced by Böhm tree equality (see (Barendregt 1984)). Similar remarks apply for various other call-by-value and lazy untyped lambda calculi.

An example which the author finds particularly appealing is the full continuous Böhm tree model. Somewhat surprisingly, this model satisfies condition (C) — the reason is explained in (Longley 1995, Section 7.4) — and so Theorem 5.13 applies. A similar observation probably holds for the "tree models" associated with other untyped lambda calculi. However, our results unfortunately do not apply to the full [resp. effective] Nakajima tree model, since this does not satisfy condition (AB) [resp. condition (B)].

Finally, it is embarrassing that two of the models of primary interest, namely $\mathcal{P}\omega$ and K_2 (along with their effective submodels) fall outside the scope of our theorems. For $\mathcal{P}\omega$, as mentioned above, the problem is that the numerals are not maximal elements; for K_2

the problem stems from the anomaly explained in Example 3.2. Of course, in both these cases the relevant results are already known, both for the standard and for the modified extensional collapse (see Section 4.2 and the remarks at the start of Section 9).

10.2. *Mild strengthenings and generalizations*

We next consider various directions in which our theorems can be extended. In this section we mention some immediate observations leading to mild strengthenings.

10.2.1. *Retracts of simple types.* In this paper we have confined our attention to the simple types over $\mathbb{N}$. However, it is well known that a rich collection of datatypes may be obtained as *retracts* of simple types (much as the simple types were seen to be retracts of pure types in Section 6.2). This allows us to add not only other base types such as the booleans, but (positive) inductive types such as lists and trees, and even dependent (sum and product) types, as well as all higher types over these. For quite general reasons, all these types can be interpreted in $\mathbf{Asm}(A)$ for any N-TPCA A, and our theorems immediately transfer to this richer class of types. (The essential ideas here were introduced in (Scott 1976); see also (Longley 2003).) Thus, we have robust notions of computability for total objects of all these types.

10.2.2. *Weaker definitions of NR-TPCA.* Firstly, whilst our definition of NR-TPCA in Section 2 seems to us to be a natural one to work with, it is clear that the full strength of this definition is not needed for our results, since we only require **rec** and **y** combinators at certain low types. In fact, in the effective case we can make do with $\mathbf{rec}_0$ and $\mathbf{y}_1$; in the continuous case $\mathbf{y}_2$ is also needed. We could therefore formulate our theorems for a weaker notion of NR-TPCA in which only these recursion operators are required. We do not know whether this would admit any particularly interesting additional examples, beyond the obvious term models. However, it is certainly of conceptual interest to know that such limited computational power suffices for computing total functionals. In the continuous case, for instance, our result shows that $\mathbf{y}_1$ and $\mathbf{y}_2$ provide all that is required over and above Kleene's schemata S1–S9 (interpreted over C) in order to generate the whole of $\mathsf{C}^{\mathit{eff}}$. This suggests that, in some sense, there are probably no *natural* notions of computability between the Kleene computable functionals on C and $\mathsf{C}^{\mathit{eff}}$ itself. (For further information on this intermediate territory, see (Gandy and Hyland 1977; Normann 1981).)

10.2.3. *Relative extensional collapses and other realizations.* For simplicity we have formulated our theorems for C and HEO in terms of a "simple" extensional collapse construction, but it is clear that the proofs also establish more general "relative versions". For instance, if A is an effective NR-TPCA (with realization γ) satisfying conditions (A) and (B) of Theorem 5.12, then any sub-NR-TPCA A' of A inherits an effective realization satisfying these conditions, and modulo some cosmetic changes, the proof of Theorem 5.12 shows that $\gamma_*(\mathbf{EC}(A;A')) \cong \mathsf{HEO}$ in $\mathbf{Asm}(K_1)$. The crucial point here is that the Normann operators $\nabla_k \in A_{\overline{1}\to\overline{k}}$ are NRComb-definable, and so lie within A'.

For the continuous case, suppose A is a full continuous NR-TPCA with realization γ. We may define a *full sub-NR-TPCA* of A to be a sub-NR-TPCA A' such for any $f : \mathbb{N} \to \mathbb{N}$ we have $f^\sharp \in A'_1$ (see Definition 5.5). If A is a full-continuous NR-TPCA satisfying conditions (AB) and (C) of Theorem 5.13, and A' is a full sub-NR-TPCA of A which moreover contains **norm**, then the proof of Theorem 5.13 goes through and shows that $\gamma_*(\mathbf{EC}(A; A')) \cong \mathsf{C}$ in $\mathbf{Asm}(K_2)$.

Carrying this line of thought a little further, we see that there is no essential reason to restrict attention to extensional collapses at all. Consider, for example, an arbitrary type-respecting realization of a total type structure T over a TPCA A — this corresponds to an internal TTS $\mathbf{T}$ within $\mathbf{Asm}(A)$. Even if $\mathbf{T}$ is not of the form $\mathbf{EC}(A)$ or $\mathbf{EC}(A; A')$, our theorems will still apply to $\mathbf{T}$ provided $\mathbf{T}$ satisfies a certain "completeness condition". For example, suppose A is an effective TPCA satisfying conditions (A) and (B) of Theorem 5.12, and suppose $\mathbf{T}$ has the following property:

— whenever $f : T_k \to \mathbb{N}$ is a total function and $\dot{f}$ is an NRComb-definable element of A_{k+1} such that $\dot{f} \cdot \dot{x} = \widehat{f(x)}$ whenever $\dot{x}$ realizes x in $\mathbf{T}_k$, there is an element of $\mathbf{T}_{k+1}$ which is realized by $\dot{f}$.

The idea is that $\mathbf{T}_{k+1}$ can comprise any class at all of "total computable functionals" $\mathbf{T}_k \to N$, provided it is rich enough to include all computations expressible in NRComb. (As discussed in Section 10.2.2, we can get away with less than the whole of NRComb here.) Under these conditions, our construction of the Normann operator $\nabla_{k+1} : \overline{1} \to \overline{k+1}$ goes through, allowing us to conclude that $\mathbf{T} \cong \mathbf{HEO}$. Analogous remarks apply in the continuous case.

This observation seems to us significantly to reinforce the impression of ubiquity suggested to our theorems, in that it vastly extends the class of realizations to which they are applicable. Moreover, it seems conceivable that there are natural approaches to *constructing* a total type structure T, using elements of some TPCA A as intensional representations, in which the elements of A_{k+1} chosen to represent total functionals need not include *all* elements of A (or some $A' \subseteq A$) that happen to behave extensionally on the chosen representatives of type k. However, we do not have any convincing examples of such constructions at present.

10.2.4. *Quotients of NR-TPCAs.* A further way to extend the scope of our theorems is by considering *quotient maps*. As a trivial example, suppose A is the applicative structure of closed terms of some deterministic simply typed programming language extending NRComb, and B is the same structure quotiented by some kind of computational equality (generated by the evaluation relation) making B an NR-TPCA. Even though A is not in general an N-TPCA, we may define its extensional collapse (relative to the PER $\sim$ at ground type, where $M \sim N$ iff M, N evaluate to the same numeral), and it is clear that $\mathsf{EC}(A)$ and $\mathsf{EC}(B)$ coincide. One may therefore directly obtain HEO (for instance) as the extensional collapse of a raw term model.

More interestingly, suppose A, B are NR-TPCAs with B a quotient of A, where the quotient map $q : A \to B$ preserves and reflects numerals (that is, x is a numeral for n in A if and only if $q(x)$ is the numeral for n in B). Again, it is clear that $\mathsf{EC}(A)$ and $\mathsf{EC}(B)$

coincide, so if the conditions of one of our theorems are applicable to either A or B, we can also deduce the result for the other. In the effective case this does not in fact lead to any new models, but in the continuous case, there is the possibility that B satisfies condition (C) while A does not.

In practice, the requirement regarding numerals turns out to be rather strong and rules out many potential applications (such as passing from the Böhm tree model to the Nakajima tree model — see Section 10.1). However, we are hopeful that it might be possible to establish the extensional collapse of the term model NRC^∞ by such means, by using an intermediate term model of which $\mathsf{NRC}^{\infty+}$ and NRC^∞ are both quotients, and appealing to normalization theorems for the relevant calculi in order to establish the numeral property. We have not worked out the details.

10.3. *Uniform versions of the Normann programs*

Central to our proofs in Sections 7 and 8 was a lemma asserting the existence of Normann programs $\mathtt{nabla}_k$ that defined suitable operators ∇_k in the structure A in question. It is therefore natural to ask whether the Normann programs can be chosen independently of A: in other words, can we give a single program $\mathtt{nabla}_k$ which works uniformly for the whole class of NR-TPCAs under consideration?

For the continuous case, the answer is immediately positive, since the programs we defined in Section 7 do not depend at all on A or on details of the realization γ. Of course, we have not been able to do without the constant `norm`, so in order to say that these programs work "uniformly", one should regard the NR-TPCAs in question as equipped with an explicit choice of an element **norm** by which `norm` must be interpreted.

In the effective case, the Normann programs in Section 8 involve a defined constant `satis-rep`, which represents a type 2 partial computable function whose definition is implicit in the proofs of Proposition 8.10 and the preceding results, and which thus depends essentially on data associated with the realization γ. Specifically, for each type level k, the function 'satis-rep' depends on choices of (a finite number of) realizers $\mathbf{a}_{\sigma\tau}$ and $\mathbf{d} \in K_1$, and on an index witnessing the computable enumerability of the set $\{m \mid m \vdash_0^\gamma \widehat{0}\}$. However, it is clear from our proofs that a suitable function 'satis-rep' is computable uniformly in these data. We may therefore bundle these parameters up into a single parameter $\wp \in \mathbb{N}$, and modify our programs so that the uniform dependence on $\wp$ is explicit.

In order to obtain a truly uniform version of the Normann programs, we may extend the trick involving the place markers e in the proof of Proposition 8.10. Specifically, we use as place markers all coded triples $\langle y, w, \wp\rangle$, where y and w as before, and $\wp$ is thought of as a potential code for the data listed above. As in Proposition 8.10, we effectively generate the tokens $\langle e, s, q\rangle$ of the satisfactory representation, now using $\wp$ as an additional parameter in a role similar to y and w. We take care to maintain condition (3) of Definition 8.9, so that even tokens arising from "bad" values of $\wp$ will be "correct" with respect to Φ. Moreover, condition (4) of Definition 8.9 will be upheld, since there is *some* value of $\wp$ which is right for the model in question, and the tokens generated using this $\wp$ will suffice to cover all possible $\dot{\lambda}$ and $\dot{G}$.

It follows from the existence of such uniform Normann programs that for any $x \in \mathsf{C}_\sigma^{\mathit{eff}}$ [resp. HEO_σ], there is some expression e_x of NRComb$^+$ [resp. NRComb] which defines a representative of x uniformly for all models satisfying the conditions of Theorem 5.14 [resp. Theorem 5.12]. In other words, whilst in general the set of expressions that denote total elements of A might fluctuate wildly as A varies, there exists a "hard core" of expressions that *always* denote total elements, and moreover every total functional has a "robust implementation" of this kind. This contributes further to the impression that our type structures enjoy a kind of stable existence; one can also imagine that the existence of these robust implementations might have interesting computer science applications.

10.4. *Prospects for further extensions and simplifications*

We would very much like to be able to weaken or dispense with the technical conditions involved in our theorems. Besides being something of a blemish from an aesthetic point of view, they restrict the applicability of our results and, as we have seen, exclude some important examples. So far we have been unable to prove our theorems under any significant weakenings of the hypotheses, despite considerable efforts in this direction. However, neither do we have any counterexamples to show that some such hypotheses are necessary.

Let us consider the situation for our theorems on the standard extensional collapse (the situation for the modified collapse being precisely analogous). In the author's view, any of the following would represent a significant advance:

— Replacing condition (B) of Theorem 5.12 with "the set of realizers for $\widehat{0}$ is Π^0_2".
— In condition (AB) in Theorem 5.14, replacing the requirement that R is open with "R is $\mathbf{\Pi}^0_2$".
— Dispensing with condition (C) in Theorem 5.13.

We are somewhat hopeful regarding the first and second of these, and have at least some ideas for possible proof strategies, though we have so far been unable to solve all the technical problems they generate. These problems seems to us to be worthy of further attention, not least because the "Π^0_2 condition" is satisfied by *all* the naturally occurring models of which we are aware.

The third relaxation would seem to be perhaps the hardest to dispense with. It is clear that our method of proof in Section 7 leans heavily on condition (C), which is used in two quite distinct roles (three in the modified version!) to eradicate unwanted information. Even so, there is a feeling that whereas something like condition (B) or (AB) is integral to our whole approach, condition (C) seems like cheating. At present we have very little idea how one might dispense with this condition in the continuous setting; however, it seems likely that any technique which would allow us to do so would also allow us to eliminate the use of the combinator $\mathtt{y}_2$.

The author's view at present is that a significant new idea would be needed to make progress on any of the above problems. Nevertheless, there remains the fact that there are *known* characterizations of our type structures via extensional collapse constructions which fall outside the scope of our theorems, which suggests that the "ubiquity" of these structures extends further than our current theorems account for.

There is also the question of whether the proofs of our current theorems can be simplified. It is tempting to wonder whether the machinery we use is more complicated than necessary for the task at hand. However, our experience suggests that our main proofs are quite delicate and finely tuned constructions — even small changes to some point of technical detail frequently have deep repercussions for the rest of the proof — so that finding substantial simplifications is unlikely to be easy. We have repeatedly found that the syntactic models NRC, NRC^{∞} and $\mathsf{NRC}^{\infty+}$ have played a valuable role in showing up problems with proposed proof strategies, and we would expect them to continue to be useful in this role for testing out any new ideas for extending or simplifying our proofs.

Even so, one may ask whether the *presentation* of our proofs might be cleaned up by means of some more high-level or abstract approach. For instance, since so much of our work has to do with the "effective content" of various mathematical statements, it is natural to ask whether parts of our arguments might be couched in the *internal logic* of the categories in question (see *e.g.* (Hyland 1982)). At present, many of our arguments (*e.g.* those involving conditions (A) and (B)) are concerned with "low-level" structure which one would not expect to be visible from within the categories themselves, and the task of isolating parts of our arguments that are amenable to an internal logic treatment would appear to be non-trivial.

A more promising possibility for abstraction (to the author's mind) comes from the work of Berger on abstract notions of density and totality in domains (Berger 1993; Berger 2002). Berger has used these ideas to obtain a domain-theoretic generalization of the KLS theorem, and we are hopeful that they may also help to clarify some of the arguments of the present paper.

11. Conclusion

We have shown that some large classes of "standard realizability" and "modified realizability" constructions all give rise to one of the well-known type structures $\mathsf{C}, \mathsf{C}^{\mathit{eff}}, \mathsf{HEO}$. As we have observed, even for individual TPCAs such results often have a non-trivial character, and so the possibility of obtaining such general results came as a considerable surprise to the author. Since our classes include a large number of mathematically natural constructions (as well as an even larger number of unnatural ones), we regard our results as providing strong evidence that these type structures are highly canonical mathematical objects. (Indeed, the interest of our theorems perhaps lies less in the source of "natural" characterizations of C, $\mathsf{C}^{\mathit{eff}}$, HEO that they yield, than in the fact that even artificial or contrived NR-TPCAs cannot avoid giving rise to these structures.) In fact, our theorems seem to us to go somewhat further and to suggest a kind of *definitive* status for these type structures, in the sense that they make it very difficult to believe that there are any other rival notions of "continuous functional on continuous data," etc., with comparably good credentials. In the author's view, this contrasts markedly with the situation for partial type structures: seemingly, the only evidence that there are no further interesting notions of computable partial functional awaiting discovery is that we have not come across them yet. (*Cf.* (Longley 2005a, Section 7).)

We venture to suggest that our theorems are not merely new results, but in some way

represent a new *kind* of result in the field of higher type computability, in that they address a whole class of constructions simultaneously. Rather than just considering a few isolated points or landmarks in the "space of notions of computability", we are able to take care of a whole tract of territory in a single swoop. We feel that the possibility of proving such general results is a positive sign for our overall project of mapping out the higher type computability landscape. It would be interesting to know whether any results in a similar vein can be obtained for partial type structures.

The results of this paper make very substantial use of the general framework of TPCAs and applicative morphisms (here in the guise of *realizations*) which we have developed in earlier works (Longley 1995; Longley 1999a; Longley 1999b), and which will be more fully presented in (Longley 2007?). The present work provides an example of a non-trivial application of these ideas — we see this as encouraging evidence of the usefulness of this framework on both a conceptual and a technical level.

Our proofs have also highlighted a curious and unexpected moral: "effective" is sometimes easier than "continuous". Admittedly, some aspects of the proof are much more subtle in the effective setting (compare Section 7.3 with Section 8.2); however, a superficial comparison of the two proofs is misleading, since condition (C) gives an unfair advantage to the continuous case. More telling is that in terms of the actual scope of our results, we have been able to get further in the effective setting. It seems that the infinitary nature of data in the full continuous setting creates genuine mathematical obstacles for certain kinds of proofs. It might be interesting to apply this lesson to other problems for which the full continuous version appears to be difficult, such as the open question of whether the "intensional" total type structure over the reals coincides with the "extensional" one (Bauer *et al.* 2002; Normann 2005). In the light of our experience it seems conceivable that, contrary to natural expectations, the effective analogue of this problem might actually turn out to be easier.

Finally, our results seem to have a bearing on a more conceptual issue. For the author, one of the most important (and nagging) problems in the subject area concerns the status of the extensional collapse construction. In the context of the present paper, the issue can be framed as follows: given that an N-TPCA A may in general support realizations of many different total type structures (and sometimes multiple non-isomorphic realizations of the same type structure), what, if anything, is special about the one picked out by the extensional collapse construction, beyond the fact that it *is* given by this construction?

On the one hand, the type structure $\mathsf{EC}(A)$ can be seen as inheriting some kind of significance from the realizability model $\mathbf{Asm}(A)$ (and in the untyped case, the associated topos), within which $\mathsf{EC}(A)$ arises from the cartesian closed structure. These categories are themselves so abundant in structure, and so rich in connections with other important ideas, as to compel attention in their own right, and this in itself would seem to be some reason for being interested in $\mathsf{EC}(A)$.

On the other hand, judged on more intrinsic criteria, it is less clear whether $\mathsf{EC}(A)$ is really special at all. In particular, the "greedy" strategy embodied by the EC construction — at each type level we include as many functionals and realizers as possible — need not yield the *unique* maximal TTS realizable over A, and there is no particular reason *a priori* to expect it to yield the "best" such TTS. (For instance, it can be argued that within

$\mathbf{Asm}(K_1)$, the internal TTS $\mathbf{C}^{\mathit{eff}}$ enjoys much better properties than **HEO**.) Moreover, the construction itself has a *prima facie* anarchic character — at each type level, one takes all realizers that simply "happen" to act extensionally on realizers of the type below, and one may feel a suspicion that the resulting structure will be unduly sensitive to "accidental" properties of the model in question — a feeling which intensifies as one climbs up the types. The same message is of course reinforced by the non-functoriality of EC, and by other anomalies such as the fact that extensional collapses do not necessarily compose — *cf.* (Honsell and Sannella 1999).

Our present results seem to us to go some way towards exorcising this sense of unease, in that they show how the wildness of the general EC construction is sometimes tameable. In particular, our "robust" NRComb implementations of total functionals (as in Section 10.3) offer a kind of common skeletal structure running through all the models in question, giving us a strong grasp on the behaviour of arbitrary extensional realizers. It is to be hoped that this structural analysis of C, $\mathsf{C}^{\mathit{eff}}$, HEO and their realizability characterizations will lead to further understanding of the associated computability notions in the future.

Acknowledgements

I would like to thank Dag Normann, firstly for developing the ideas on which the present paper is based, and secondly for illuminating correspondence and discussions. I have also profited from discussions with Andrej Bauer, Ulrich Berger, Martín Escardó, Gordon Plotkin and Alex Simpson. At a somewhat earlier stage, Martin Hyland helped me considerably in my understanding of the structures considered in this paper.

I am also grateful to the editors for their persistence in encouraging me to write this paper, and their patience in waiting so long for me to do so. The two anonymous referees took a great deal of care over the paper, spotting numerous minor errors and offering valuable suggestions for improvement. Finally, I thank Caroline Pilkington (now Longley) for her unfailing support and encouragement, and for cheerfully tolerating my daily progress reports from the "mathematical ridge".

References

Abramsky, S., and G. McCusker, *Game semantics*, in Schwichtenberg, H., and U. Berger, editors, "Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School", Springer (1999), 1–56.

Amadio, R., and P.-L. Curien, "Domains and Lambda Calculi", Cambridge Tracts in Theoretical Computer Science **46**, Cambridge University Press (1998).

Awodey, S., L. Birkedal, and D.S. Scott, *Local realizability toposes and a modal logic for computability*, Mathematical Structures in Computer Science **12** (2002), 319–334.

Barendregt, H.P., "The Lambda Calculus: Its syntax and semantics" (revised), Studies in Logic and the Foundations of Mathematics **103**, North-Holland (1984).

Bauer, A., "The Realizability Approach to Computable Analysis and Topology", Ph.D. thesis, Carnegie Mellon University (2000). Available as technical report CMU-CS-00-164.

Bauer, A., *A relationship between equilogical spaces and Type Two Effectivity*, Mathematical Logic Quarterly **48(1)** (2002), 1–15.

Bauer, A., M. Escardó, and A. Simpson, *Comparing functional paradigms for exact real-number computation*, in Proceedings of ICALP 2002, Springer LNCS **2380** (2002), 488–500.

Beeson, M., "Foundations of Constructive Mathematics", Springer-Verlag (1985).

Berger, U., *Total sets and objects in domain theory*, Annals of Pure and Applied Logic **60** (1993), 91–117.

Berger, U., *Computability and totality in domains*, Mathematical Structures in Computer Science **12** (2002), 281–294.

Bergstra, J.A., *The continuous functionals and* ^2E, in Fenstad, J.E., R.O. Gandy, and G.E. Sacks, editors, "Generalized Recursion Theory II", North-Holland (1978), 39–53.

Berry, G., *Stable models of typed lambda-calculi*, in Proceedings of 5th International Colloquium on Automata, Languages and Programming, Springer LNCS **62** (1978), 72–89.

Berry, G., and P.-L. Curien, *Sequential algorithms on concrete data structures*, Theoretical Computer Science **20(3)** (1982), 265–321.

Bezem, M., *Isomorphisms between HEO and* HRO^E*, ECF and* ICF^E, Journal of Symbolic Logic **50** (1985), 359–371.

Bucciarelli, A., and T. Ehrhard, *Sequentiality and strong stability*, in Proceedings of 6th Annual Symposium on Logic in Computer Science, IEEE (1991), 138–145.

Ershov, Yu.L., *Computable functionals of finite types*, Algebra i Logika **11(4)** (1972), 367–437. English translation in Algebra and Logic **11(4)**, 203–242 (AMS).

Ershov, Yu.L., *Everywhere-defined continuous functionals*, Algebra i Logika **11(6)** (1972), 656–665. English translation in Algebra and Logic **11(6)**, 363–368 (AMS).

Ershov, Yu.L., *Maximal and everywhere defined functionals*, Algebra i Logika **13(4)** (1974), 374–397. English translation in Algebra and Logic **13(4)**, 210–225 (AMS).

Ershov, Yu.L., *Hereditarily effective operations*, Algebra i Logika **15(6)** (1976), 642–654. English translation in Algebra and Logic **15(6)**, 400-409 (AMS).

Ershov, Yu.L., *Model C of the partial continuous functionals*, in Gandy, R.O., and J.M.E. Hyland, editors, "Logic Colloquium 1976", North-Holland (1977), 455–467.

Escardó, M., J. Lawson, and A. Simpson, *Comparing cartesian closed categories of (core) compactly generated spaces*, Topology and its Applications **143** (2004), 105–145.

Gandy, R.O., *Effective operations and recursive functionals (abstract)*, Journal of Symbolic Logic **27** (1962), 378–379.

Gandy, R.O., and J.M.E. Hyland, *Computable and recursively countable functions of higher type*, in Gandy, R.O., and J.M.E. Hyland, editors, "Logic Colloquium 1976", North-Holland (1977), 407–438.

Hinata, S., and S. Tugué, *A note on continuous functionals*, Annals of the Japan Association for Philosophy of Science **3** (1969), 138–145.

Honsell, F., and D. Sannella, *Pre-logical relations*, in Computer Science Logic 1999, proceedings, LNCS **1683**, Springer (1999), 546–561.

Hyland, J.M.E., "Recursion theory on the countable functionals", Ph.D. thesis, University of Oxford (1975).

Hyland, J.M.E., *Filter spaces and continuous functionals*, Annals of Mathematical Logic **16** (1979), 101–143.

Hyland, J.M.E., *The effective topos*, in Troelstra, A.S. and D. van Dalen, editors, "The L.E.J. Brouwer Centenary Symposium", NorthHolland (1982), 165–216.

Hyland, J.M.E., *Game semantics*, in Pitts, A.M., and P. Dybjer, editors, "Semantics and Logics of Computation", Cambridge University Press (1997), 131–194.

Kleene, S.C., "Introduction to Metamathematics", Wolter-Noordhoff and North-Holland (1952).

Kleene, S.C., *Recursive functionals and quantifiers of finite types I*, Transactions of the American Mathematical Society **91** (1959), 1–52.

Kleene, S.C., *Countable functionals*, in Heyting, A., editor, "Constructivity in Mathematics: Proceedings of the Colloquium held at Amsterdam, 1957", North-Holland (1959), 81–100.

Kleene, S.C., and R.E. Vesley, "The Foundations of Intuitionistic Mathematics", North-Holland (1965).

Kreisel, G., *Interpretation of analysis by means of functionals of finite type*, in Heyting, A., editor, "Constructivity in Mathematics: Proceedings of the Colloquium held at Amsterdam, 1957", North-Holland (1959), 101–128.

Kreisel, G., D. Lacombe, and J.R. Shoenfield, *Partial recursive functionals and effective operations*, in Heyting, A., editor, "Constructivity in Mathematics: Proceedings of the Colloquium held at Amsterdam, 1957", North-Holland (1959), 290–297.

Lietz, P., and T. Streicher, *Impredicativity entails untypedness*, Mathematical Structures in Computer Science **12** (2002), 335–347.

Longley, J.R., "Realizability Toposes and Language Semantics," Ph.D. thesis, University of Edinburgh, 1995. Available as technical report ECS-LFCS-95-332.

Longley, J.R., *Matching typed and untyped realizability*, Electronic Notes in Theoretical Computer Science **23(1)** (1999).

Longley, J.R., *Unifying typed and untyped realizability*, Unpublished note, 1999. Available at `http://www.inf.ed.ac.uk/home/jrl/unifying.txt`.

Longley, J.R., *The sequentially realizable functionals*, Annals of Pure and Applied Logic **117(1)** (2002), 1–93.

Longley, J.R., *Universal types and what they are good for*, In Zhang, G.-Q. et al., editors, "Domain theory, Logic and Computation: Proceedings of 2nd International Symposium on Domain Theory, Sichuan, China, 2001", Kluwer (2003), 25–63.

Longley, J.R., *Notions of computability at higher types I*, in Cori, R., A. Razborov, S.Todorčević and C. Wood, editors, "Logic Colloquium 2000: Proceedings of the ASL meeting held in Paris", Lecture Notes in Logic 200, ASL (2005), 32–142.

Longley, J.R., *On the ubiquity of certain total type structures*, extended abstract, in "Proceedings of the Workshop on Domains VI, Birmingham, UK", Electronic Notes in Theoretical Computer Science **73**, Elsevier (2005), 87–109.

Longley, J.R., *Notions of computability at higher types II*, in preparation.

Milner, R., *Fully abstract models of typed λ-calculi*, Theoretical Computer Science **4** (1977), 1–22.

Milner, R., M. Tofte, R. Harper, and D. MacQueen, "The Definition of Standard ML (revised)". MIT Press (1997).

Moggi, E., *Computational lambda-calculus and monads*, Information and Computation **93(1)** (1991).

Normann, D., "Recursion on the countable functionals", Springer Lecture Notes in Mathematics 811 (1980).

Normann, D., *The continuous functionals: computations, recursions and degrees*, Annals of Pure and Applied Logic **21** (1981), 1–26.

Normann, D., *The continuous functionals*, in Griffor, E.R., editor, "Handbook of Computability Theory", North-Holland (1999), 251–275.

Normann, D., *Computability over the partial continuous functionals*, Journal of Symbolic Logic **65** (2000), 1133–1142.

Normann, D., *Comparing hierarchies of total functionals*, Logical Methods in Computer Science **1(2)** (2005).

Normann, D., E. Palmgren, and V. Stoltenberg-Hansen, *Hyperfinite type structures*, Journal of Symbolic Logic **64** (1999), 1216–1242.

Oosten, J. van, *The modified realizability topos*, Journal of Pure and Applied Algebra **116** (1997), 273–289.

Oosten, J. van, *A combinatory algebra for sequential functionals of finite type*, in Cooper, S.B., and J.K. Truss, editors, "Models and Computability", Cambridge University Press (1999), 389–406.

Platek, R., "Foundations of recursion theory," Ph.D. thesis, Stanford University (1966).

Plotkin, G., *Set-theoretical and other elementary models of the λ-calculus*, Theoretical Computer Science **121** (1993), 351–409. First written in 1972 and circulated in unpublished form.

Plotkin, G.D., *LCF considered as a programming language*, Theoretical Computer Science **5** (1977), 223–255.

Plotkin, G.D., $\mathbb{T}^\omega$ *as a universal domain*, Journal of Computer and System Sciences **17** (1978), 209–236.

Plotkin, G.D., *Full abstraction, totality and PCF*, Mathematical Structures in Computer Science **9(1)** (1997), 1–20.

Schwichtenberg, H., *Density and choice for total continuous functionals*, in Odifreddi, P., editor, "Kreiseliana: About and around Georg Kreisel", A.K. Peters (1996), 335–362.

Scott, D.S., *A type-theoretical alternative to ISWIM, CUCH, OWHY*, Theoretical Computer Science **121** (1993), 411–440. First written in 1969 and circulated in unpublished form since then.

Scott, D.S., *Continuous lattices*, in Lawvere, F.W., editor, "Toposes, Algebraic Geometry and Logic", Springer (1972), 97–136.

Scott, D.S., *Data types as lattices*, SIAM Journal of Computing **5(3)** (1976), 522–587.

Soare, R.I., *The history and concept of computability*, in Griffor, E.R., editor, "Handbook of Computability Theory", North-Holland (1999), 3–36.

Stoltenberg-Hansen, V., I. Lindström, and E.R. Griffor, "Mathematical Theory of Domains", Cambridge Tracts in Theoretical Computer Science **22**, Cambridge University Press (1994).

Troelstra, A.S., "Metamathematical Investigation of Intuitionistic Arithmetic and Analysis", Lecture Notes in Mathematics **344**, Springer-Verlag, 1973.

Weihrauch, K., "Computability", EATCS Monographs on Theoretical Computer Science **9**, Springer-Verlag, 2000.